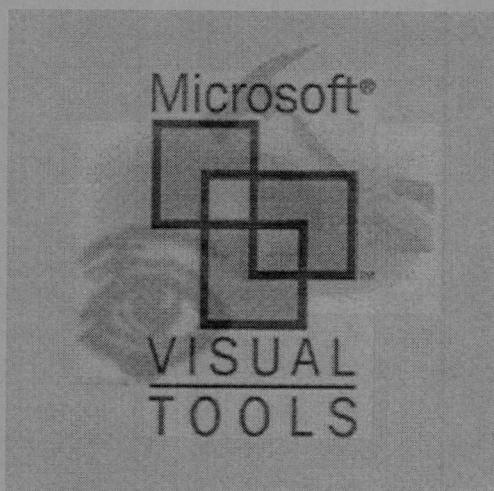


.....



Microsoft Developer
Seminar-in-a-Box Series

Visual FoxPro 3.0
Seminar Attendee
Workbook

Microsoft, Windows, and Win32, are registered trademarks and Visual FoxPro is a trademark of Microsoft Corporation. All other trademarks, marked and not marked, are property of their respective owners.

This document is provided for informational purposes only. The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to change in market conditions, it should not be interpreted to be a commitment on the part of Microsoft and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

INFORMATION PROVIDED IN THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND FREEDOM FROM INFRINGEMENT. The user assumes the entire risk as to the accuracy and the use of this document. This document may be copied and distributed subject to the following conditions: 1) All text must be copied without modification (except foreign language translation) and all pages must be included; 2) All copies must contain Microsoft's and Application Developer Training Company's copyright notice and any other notices provided therein; and 3) **This document may not be distributed within the boundaries of Canada or the United States of America.**

Copyright © 1996 Application Developers Training Company. All Rights Reserved.

Welcome to Visual FoxPro 3.0

Visual FoxPro 3.0 represents the biggest shift in development ever seen in the Xbase market. Microsoft spent a great deal of time and effort to create the greatest database development tool ever. This section describes some of the biggest changes found in Visual FoxPro 3.0

Design Tools

The biggest change in Visual FoxPro comes in the form of a new set of design tools. Microsoft's primary goal in developing Visual FoxPro was to provide a set of powerful, easy-to-use, and consistent design tools. Some of the new design tools and features include:

An Improved Project Manager

The first big change in the Visual FoxPro design tools is the Project Manager. The new project manager provides a set of convenient tools for managing the components of an application. This new project manager combines a File Manager style interface with a tabbed dialog that eases the access to your application components.

Along with simplifying the access to your components, the project manager features drag and drop capabilities. You can drag and drop components directly from the Project Manager onto your forms and reports. Development could not get any easier.

A Form Designer

Probably the biggest change relating to the Visual FoxPro environment comes in the Form Designer. The Visual FoxPro form designer includes new controls that you can immediately begin using on your forms. These controls include: a combo box, a tabbed dialog control, timers, and an integrated browse window (the grid).

Along with this set of tools, Visual FoxPro includes a set of powerful development dialogs including: alignment tools, a color palette, and a properties sheet. Also, you no longer need to write the code to open and close your tables; your forms can control access to their tables with a new tool known as the Data Environment.

An Object Oriented Class Designer

You no longer have to reinvent the wheel in your Visual FoxPro applications because Visual FoxPro is now object oriented. Object oriented development is no longer just a buzzword for selling the latest development tools. Using object oriented development techniques, you can improve your development performance by a factor of ten.

Visual FoxPro provides a set of object oriented development tools that you can use to create re-usable components that can be dropped into your development environment immediately. Visual FoxPro supports all of the OOP buzzwords: encapsulation, polymorphism, and most importantly inheritance.

An Integrated Data Dictionary

At long last Visual FoxPro includes an integrated data dictionary. You no longer need to worry about some rampant application corrupting your data. Visual FoxPro now enforces data integrity at the engine level. You can create default values for fields, validate fields, validate records before they are written, and enforce referential integrity with the new Referential Integrity Wizard. Your data is protected with Visual FoxPro's integrated data dictionary.

A Report Writer

Finally, Visual FoxPro has a new and improved set of report writing tools. You can zoom in and out on your report multiple levels, you can call procedures before and after a band has been entered, and you can have reports open and close their files using the Report Data Environment.

Architecture Changes

Along with an advanced set of development tools, Visual FoxPro includes a new and refined database development architecture. Things that were hard in some other Xbase languages are now easy in Visual FoxPro. Some of the new architectural changes include:

Windows Style Event Model

All FoxPro 2.x developers know that creating applications that behave like other Windows applications is quite a task. How many days did you fight to get your Foundation READ event loop to work correctly? How many of you even know what a Foundation READ is? Now, you no longer have to worry about foundation READ. Your Visual FoxPro forms know how to behave in

the Windows environment. They can respond to events like: Activate, Deactivate, Load and Resize.

The ability to control your applications using commonly known Windows events greatly simplifies your development day. Now you can concentrate your time on developing an application, rather than spending a great deal of time on mundane tasks, like clauses found in the READ command.

Object Oriented Development Model

Now that Visual FoxPro has gone Object Oriented, you can create applications using FoxPro's set of built in components. Or better yet, you can create a set of custom development components for use in your applications. Some of the items available to you include:

- Built-In Classes
- Properties
- Events
- Methods

The Project Manager

Your First Steps to Application Development

The first steps to developing an application are usually the hardest. The way that you organize and develop your applications can greatly affect the quality and satisfaction of the development process. Visual FoxPro provides a tool that greatly simplifies the steps necessary to begin developing your first applications. This tool is called the Project Manager.

In Visual FoxPro, you use the Project Manager to organize the components of your applications into a single cohesive development environment. This tool becomes the central repository for all of your application components and information.

Prior to using the Project Manager you should organize your development environment. Applications of all sizes consist of large numbers of individual components. These components include forms, reports, labels, programs, and others. When developing an application, it is wise to provide some level of organization for these components. A common method used to organize components is to place them in directories by component type. Below is a commonly used set of directories that can be used to organize the components of an application:

```
PROJECTNAME * Project file goes here
DATA        * Database files go here
FORMS       * Forms go in this directory
PROGRAMS    * Program files
REPORTS     * Reports (.FRX files)
COMMON      * This holds your function/class libraries
FLLS        * VFP API files
SCRATCH     * Any temporary files created by your app
MENUS       * Menu files
BITMAPS     * Bitmap files
```

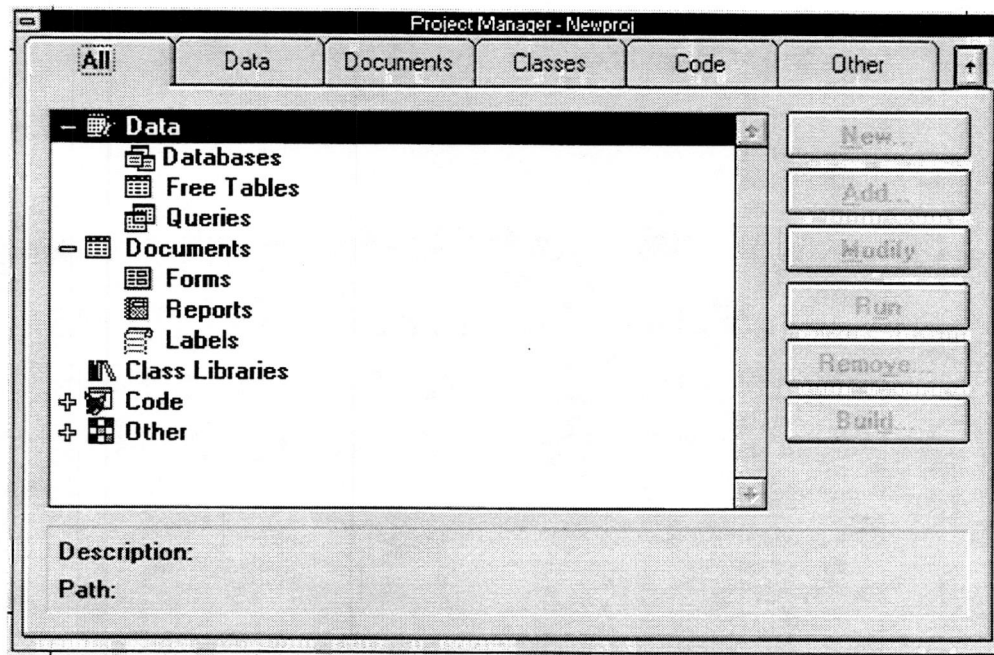
Upon creating this structure you are ready to begin creating your first project.

Creating a Project

The first step in creating a new Visual FoxPro application is to create a new project. You can create a project using two different methods. The first method is to select File-New from the Visual FoxPro menu. Upon selecting this option Visual FoxPro allows you to create a new project from a provided dialog. The second method is to type the following command in the command window:

```
CREATE PROJECT <project name>
```

Regardless of the method you use, the following empty project window appears:



At the same time, FoxPro adds a Project menu pad to the system menu bar. You make several important settings from the Option option of this menu. Click on this option and the Options dialog box appears.

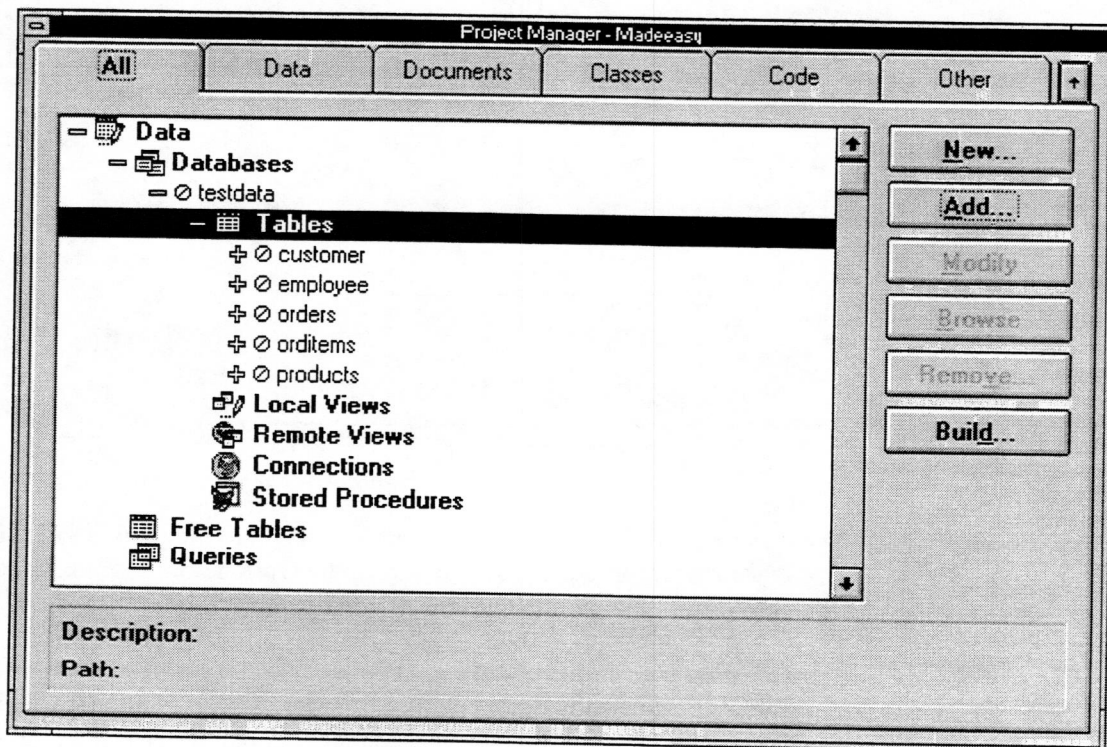
After creating your project, you can begin adding the components for your application. You can add components a number of ways. The first way is to click the Add button from the Project Manager container. The Add button allows you to select from files that have been created previously. Upon clicking the add button Visual FoxPro prompts you with a Windows File Open dialog allowing you to add the appropriate type of file.

The second method of adding a file to the Project Manager is to create a new file for your application. You can create a new file by clicking the New button from the Project Manager container. Upon selecting New from the Project Manager, Visual FoxPro will prompt you for the name of the new file. After this information is entered, you can create components using the appropriate design tools.

The following exercise shows you how to add an existing database to your project:

1. Create a new project if you have not already done so.
2. Move the Project Manager to the Data outline option. Click on the plus (+) icon; this opens the data sub-outline.
3. Move your cursor to the Databases outline option.
4. Click the Add button in the project container.
5. Select your database using the File-Open dialog provided by Visual FoxPro.

The following illustration shows what your project will look like upon adding a database:



Setting Project Options

The project manager allows you to set specific project options related to the generation of code, icons, and project owner information. To specify project options, select Project-Project Info from the Visual FoxPro menu. Selecting this option presents you with the following dialog:

The screenshot shows a Windows-style dialog box titled "Project Information - Newproj". It has two tabs: "Project" and "Files". The "Project" tab is active and contains the following fields and controls:

- Author:** A text input field.
- Company:** A text input field.
- Address:** A text input field.
- City:** A text input field.
- State:** A text input field.
- Country:** A text input field.
- Postal Code:** A text input field.
- Home:** A text input field containing "c:\vfp" and a browse button ("...").
- Last Built:** A text input field.
- Files:** A text input field containing "0".
- Checkboxes:**
 - ☒ **Debug Info**
 - ☐ **Encrypted**
 - ☐ **Attach Icon to .EXE**
- Icon:** A button labeled "Icon..."

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

The Project Manager (PM) options are split into sub-options in separate boxes within this screen. The Developer Information is used by the different generators to create copyright text in the generated code. The Project Information contains three separate check boxes, that you use to specify whether debugging information is included, the finished code is encrypted, and the FoxPro logo is enabled at startup.

By checking the Include Debugging Information box, you can follow your code, line by line, in the FoxPro trace window during debugging exercises. Including debugging information is a good idea during development, but should be unchecked when building the distributed version of your application because the resulting .APP or executable is smaller.

The Encrypt option refers to the enabling or disabling of encryption of the .APP or executable file. FoxPro uses this method to scramble compiled code for security purposes. This is necessary because unencrypted FoxPro object code is relatively easy to decompile back into source code. Utilities for this purpose are commercially available.

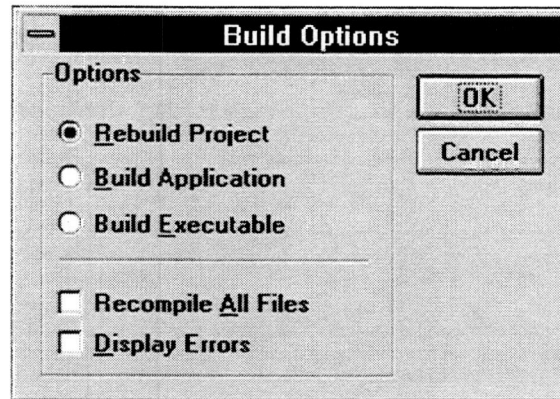
Encryption is especially useful for distributed applications and gives the developer an added level of security. Encryption does not affect the execution speed of applications, but there is a slight penalty in load time the first time each module is executed. This is usually not noticeable except with very large components (a program, for example, that approaches the FoxPro maximum compiled size of 64k) on very slow hardware.

Even encryption is not airtight; like most encryption schemes, FoxPro's has been broken. However, encryption is a complicating factor for the would-be code pirate by making decompiling much slower.

The Home field tells PM the location of the main file for your application. You can then add other directories under the home directory to include your other components. To move a project to another home directory, be sure to duplicate the sub-directory structure that exists under the current project home directory. Otherwise, the PM cannot locate the remaining components of your application.

Creating Your First Application

Upon adding all of the components to your application you can compile all of these components into a single application ready for distribution. To do this, select the Build option from the Project Manager container. Selecting this option prompts you with a dialog similar to the one below:



This dialog allows you to specify the compilation options for your project. The first set of options tell Visual FoxPro how to compile your application. These options are:

1. *Rebuild Project*: This option recompiles any programs, class libraries or screens that have been changed. It also attempts to resolve any problems that the project might have, such as files not being found.
2. *Build Application*: This option performs the same steps as the Rebuild Project option and also creates a file with the .APP extension. This file consists of all of the compiled components specified in the project manager.
3. *Build Executable*: If you own the Professional Edition of Visual FoxPro, this option is available to you. This option performs the steps performed in the Rebuild Project action and also creates a Windows Executable file (.EXE). This file can be redistributed royalty free to your users and customers. You learn about distributing .EXE files later in this seminar.

The other two options found in this dialog are Recompile All Files and Display Errors. Recompile All Files recompiles all of your project's screens, programs, and class libraries. It is a good idea to use this option before creating an .EXE file as it will reduce the size of the .EXE file. The second option, Display Errors, tells Visual FoxPro to display any errors it found in any of the compiled programs. If this option is not specified, Visual FoxPro saves these errors in a file with an .ERR extension.

Now that you have learned how to create and configure the options for a project, you can begin creating the individual components for your application. The remainder of this seminar is spent creating these components and adding them to your application.

Using the Data Dictionary

Objectives

In this section you learn how to:

- Enforce validation in your applications using Visual FoxPro's Data Dictionary.
- Set relationships between tables
- Create database triggers.
- Implement data buffering in your applications.

The Data Dictionary

Creating a robust application takes a lot of time and effort. The most important aspect of a database application is how well it manages the creation and revision of its data. This section describes the tools found in the Visual FoxPro data dictionary and how to use them successfully in your applications.

Key Terms

Before you read about the Visual FoxPro Data Dictionary, review a few key terms:

Basic Database Terminology

Database	A collection of related information about a particular topic. Usually a collection of tables.
Table	Stores information about particular entities or groups of entities. Common entities include employees, customers, and inventory items.
Row (Record)	Stores information about individual entities. Each row is broken into components called columns.
Column (Field)	An attribute of a single entity. For example, a customer has a city and a phone number. City and phone number become columns in your customer table.
Data Type	The type of data stored in a particular column. Common data types include character data, dates, and numeric data.
Index	A logical ordering of a column or combination of columns found in a table.

What Is a Data Dictionary?

Data Dictionaries are repositories that store descriptive information about your databases. This information includes names of tables, relationships between tables, integrity rules, and business rules.

Visual FoxPro's Data Dictionary is an active data dictionary. This means that if business rules are defined for your table(s), they are enforced whenever these tables are open. Some of the business and integrity rules that Visual FoxPro can enforce include:

- Default field values
- Field validation rules
- Table relationships
- Database triggers

What Does the Data Dictionary Mean To Visual FoxPro Developers?

In most Xbase applications, enforcement of database integrity occurs at the application level instead of at the database level. This means that each time data is added to your databases, you are required to validate the data in your application before writing it to the database. If you forget to do a validation in your database, your Xbase applications will not alert you that a validation error has been made. It simply writes the data to the database.

Visual FoxPro lets you enforce data integrity at the database level. Visual FoxPro's Data Dictionary does not allow you to skip the validation in your application—it merely prevents you from making inadvertent programming mistakes.

Using the Data Dictionary

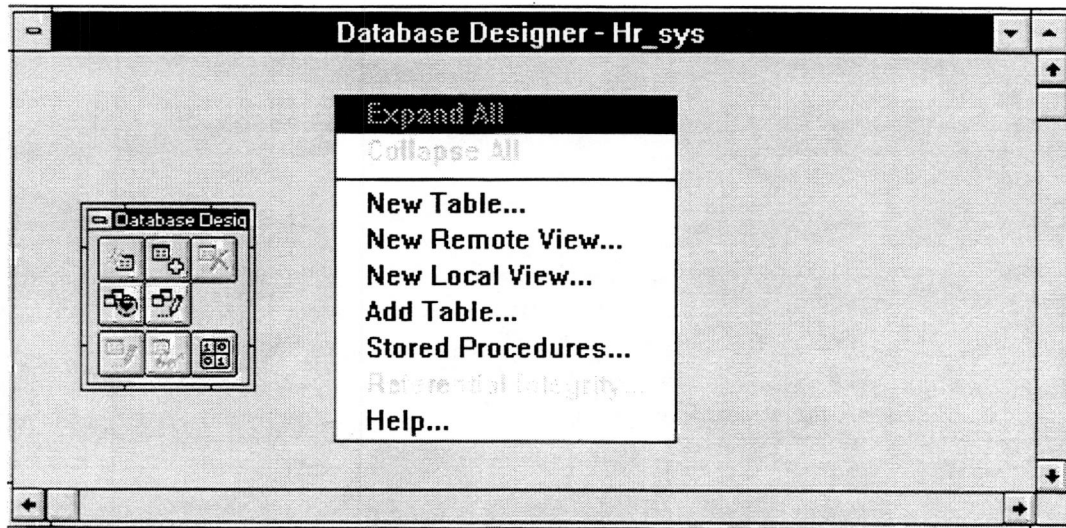
Your first step in using the Visual FoxPro Data Dictionary is to create a new database. Select the Database option from the Project outline, and then select New. Or, type the following command in the Visual FoxPro Command Window:

```
CREATE DATABASE <databasename>
```

If you create the database from the Command Window, follow the CREATE DATABASE command with a MODIFY DATABASE command. For example, if you want to create a database named HR_SYS from the Command Window, issue the following commands:

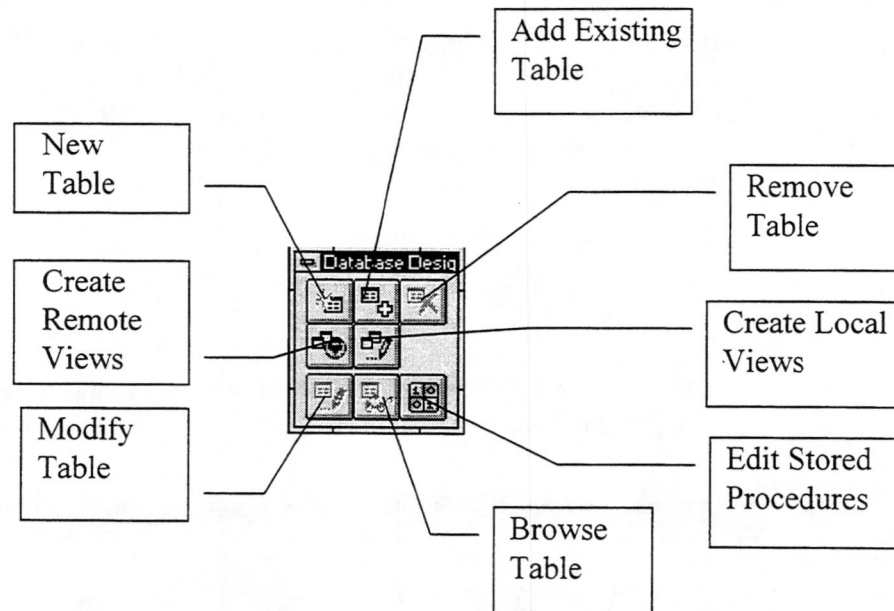
```
CREATE DATABASE HR_SYS  
MODIFY DATABASE
```

Visual FoxPro then opens the Database Designer tool shown below. The popup menu is activated by right-clicking in the Database Designer window.



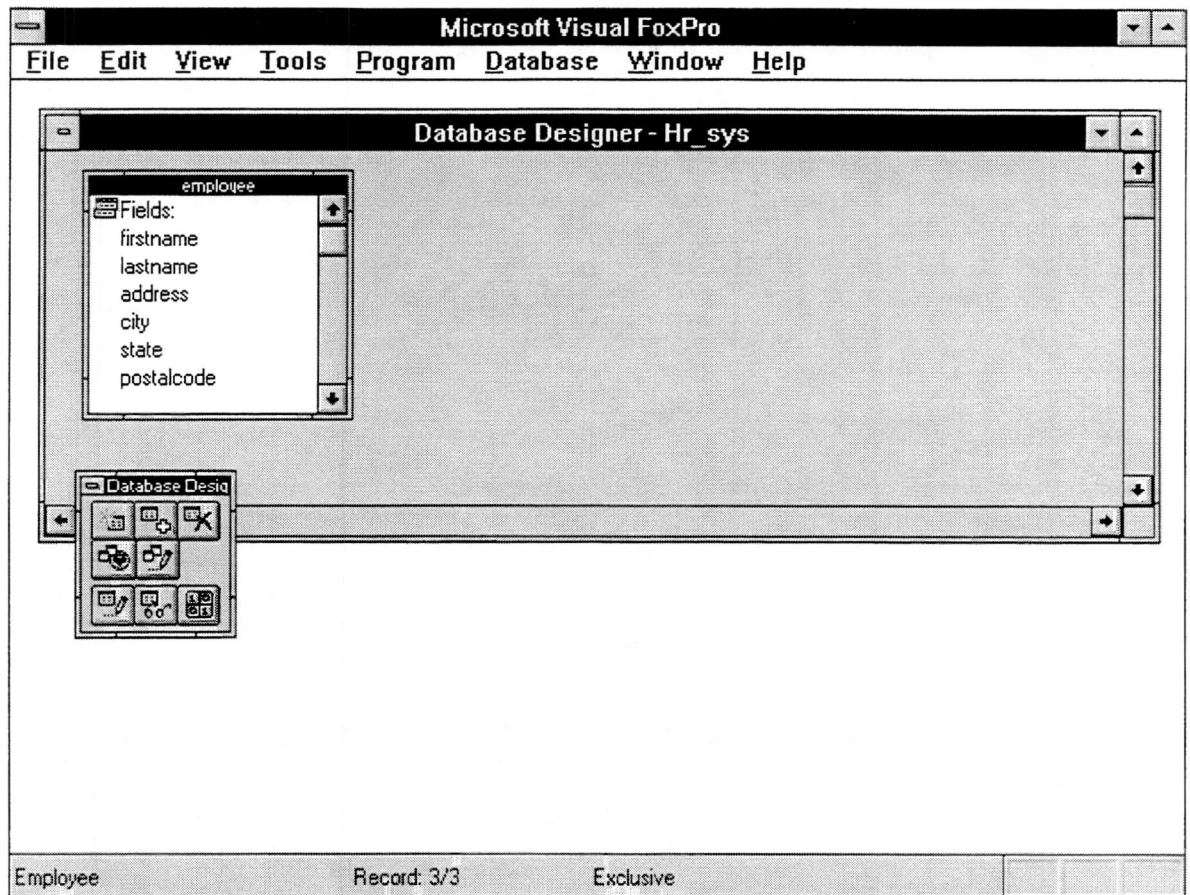
To effectively use the Database Designer, you need to understand its design components. The first component is the Database Designer Toolbar.

The *Database Designer Toolbar* allows you to maintain many of the components contained in your databases. The illustration below highlights the functions of the Database Designer Toolbar.



The first thing you need to do to your database is add the tables that belong to the database. One way to do this is to select the Add Table option from the Database Designer Toolbar. The second way to add tables is to right-click on the Database Designer design surface and activate the Database Designer popup menu. This popup menu allows you to perform any activity available in the Database Designer Toolbar options.

If you select Add Table from this dialog, Visual FoxPro prompts you with a Windows open file dialog. Next, simply select the table to add to your Visual FoxPro Data Dictionary. The figure below shows the addition of an employee table to your HR_SYS database:



After you add existing tables to the database, you can create new tables by selecting Add Table from either the Database Designer Toolbar or the Database Designer popup dialog. When you choose to add a table, Visual FoxPro pops up the Table Designer dialog.

After you add tables to your Visual FoxPro Data Dictionary, you can begin adding validations. Visual FoxPro's Data Dictionary allows you to create two types of data validations: table level validations and field level validations.

The following are the table level validations:

- Table triggers
- Table level validation code

For each field in a table you can define the following items:

- Long field names
- Validation rules
- Validation text
- Default values
- Captions
- Field comments

You can define these items from the Table Designer dialog box. The following illustration shows the Table Designer with the available rules for a selected table:

Table Designer

Table Name: Database:

Table		Index	
Name	Type	Width	Decimal NULL
↕ cno	Character	5	
company	Character	35	
contact	Character	20	
address	Character	30	
city	Character	15	
state	Character	2	
zip	Character	5	

Field Properties (cno)

Validation Rule: ...

Validation Text: ...

Default Value: ...

Caption: ...

Field Comment:

Defining a Validation Rule

The *Validation Rule* field allows you to define an expression that is evaluated any time the specified field is changed in the table. This expression is commonly a user-defined function.

If the evaluated expression is false (.F.), the field changes cancel out, and the text defined in the Validate Text clause is displayed in an error message dialog. If the evaluated expression is true (.T.), the changes are allowed in the field.

Defining a Default

One powerful feature of the Data Dictionary is its ability to define defaults for fields. The *Default Value* is an expression that Visual FoxPro evaluates and stores into the table when a record is added. If a default value is not specified Visual FoxPro inserts either a blank value or a .NULL. value, if null support is enabled for that field.

Defining Captions and Comments

The last two options you can define for the fields in a Visual FoxPro table are the *Caption* and the *Comment* fields.

The Caption expression is either a character string, or an expression that returns a character string. It is displayed as a field header for grids or browse windows.

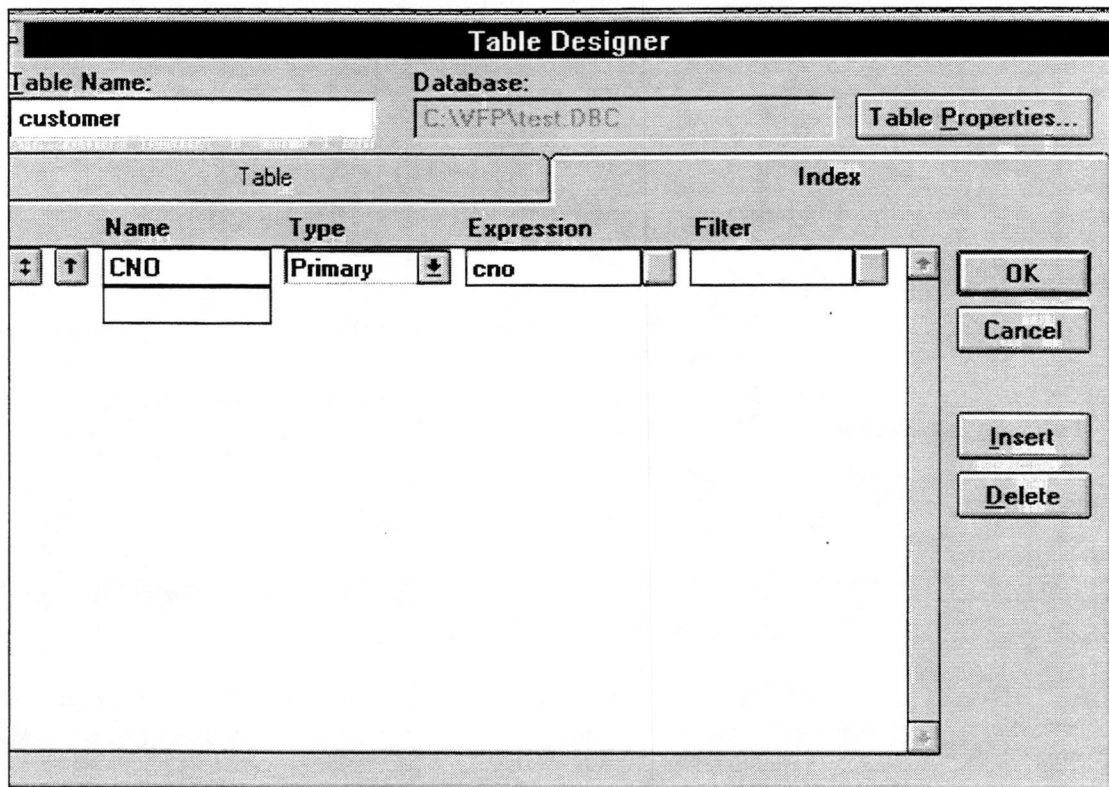
The Comments option allows you to specify alphanumeric comments for each column definition. You can place any quantity of comments in each field definition. The limit is the amount of space available on your hard drive. Whenever possible, specify comments for your column definitions. You can refer to them long after you have developed your systems.

NOTE: The first line of comments is displayed in the description area of the Project Manager. It therefore may be advantageous to keep the first line of a comment brief and descriptive.

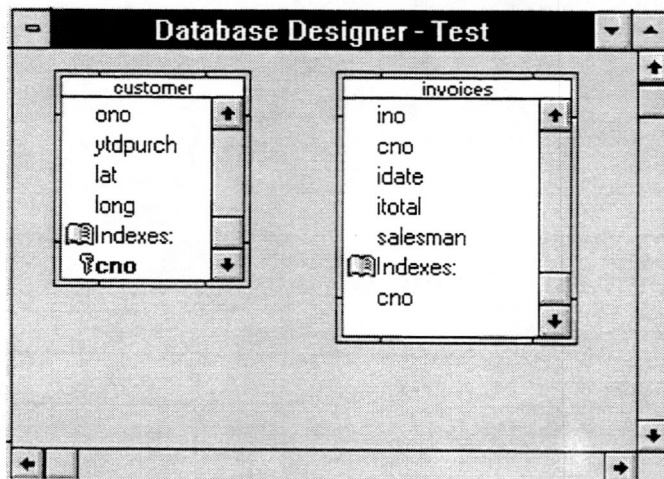
Creating Relationships Between Tables

Visual FoxPro's Data Dictionary allows you to define relationships between tables that are preserved at the data dictionary level. After you define the tables for your database, you can establish relationships between any tables found in the data dictionary. Before you can establish a relationship, however, Visual FoxPro requires that the tables to be related have indexes.

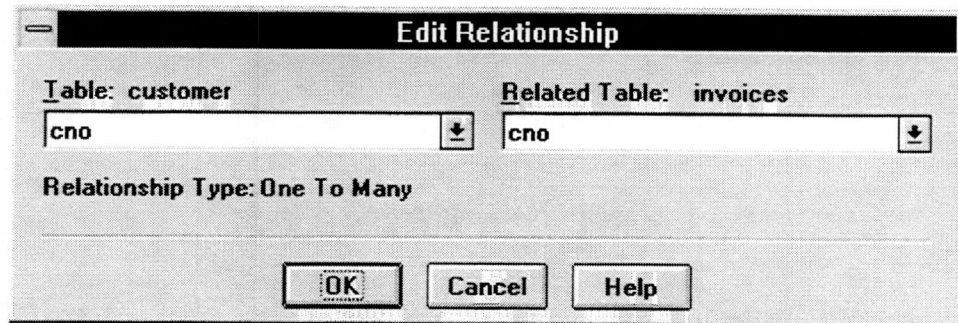
To define a one-to-many relationship, the parent table must have a primary key, and the child record must have a traditional FoxPro-style index. A regular style of index permits duplicate key values, which are required for a one-to-many type of relationship. The following screen shows how to create a primary key using the Index tab of the Table Designer:



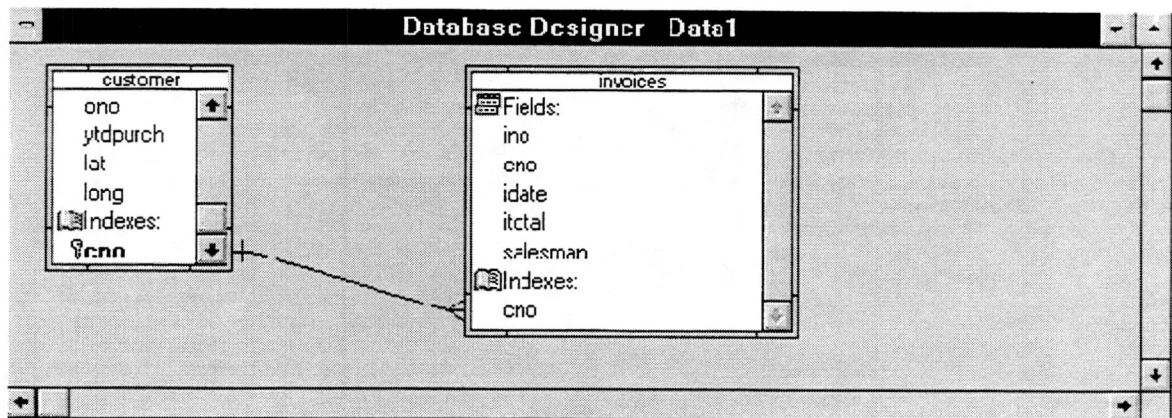
After you create the correct indexes, drag the related field from the parent table to the child table. The following screen shows the database container prior to setting a relationship:



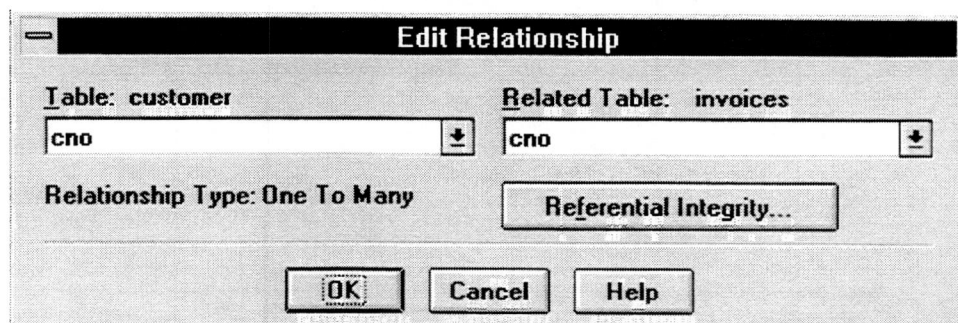
Next, Visual FoxPro pops up the following dialog box:



This screen allows you to specify the fields that the relationship uses. After you establish a relationship between two tables, Visual FoxPro automatically relates your tables whenever you use the related tables in reports or forms. After establishing the relationship, your data dictionary looks like this (Note that the crow's feet on the invoices table signify the many side of the established relationship.):



After you relate two tables, double-click on the relations bar and the *Edit Relationship* dialog box pops up. This is the same dialog box that Visual FoxPro presented when you initially established a relationship, with one addition. This dialog box now has a Referential Integrity button:



Referential Integrity is a set of rules that maintains relationships between files. The primary objective of Referential Integrity is to prevent the creation of

orphan records. Orphan records are child records without corresponding parents.

When you click the Referential Integrity button, Visual FoxPro activates the Referential Integrity Builder:

Parent Table	Child Table	Update	Delete	Insert	Parent Tag	Child Tag
customer	invoices	Ignore	Ignore	Ignore	cno	cno

Rules for Updating **Rules for Deleting** Rules for Inserting

Which rule do you want to apply when a record in the parent table is deleted?

☐ Cascade: Delete all related records in the child table.
☐ Restrict: Prohibit the deletion if there are related records in the child table.
☒ Ignore: Allow the deletion and leave related records in the child table alone.

OK Cancel Help

Define Cascading Abilities

The Referential Integrity Builder allows you to create two kinds of Referential Integrity rules: Restrictive and Cascading.

- *Restrictive Integrity* prevents you from adding data when its addition would violate a relationship. (A child without a parent.)
- *Cascading Integrity* sets up your database so that whenever you delete a parent record, you delete the child records with it.

The Referential Integrity Builder creates a set of database triggers (defined in the following paragraph) that are to be used by your database to enforce this integrity.

Creating Database Triggers

Visual FoxPro's Data Dictionary has the ability to create *Triggers*. Triggers are programs that are executed whenever a developer inserts, updates, or deletes records. Triggers are different than field level validations. Triggers are table level validations that fire whenever a specific action (insert, update, delete) happens. Field level validations only occur when the specified field is changed by the user or by the program. Field level validations fire before table level validations, so field validations happen before triggers.

Use triggers to accomplish the following:

- Implement Referential Integrity.
- Update tables automatically based on changes to specific fields.
- Validate data prior to saving changes to disk.

In Visual FoxPro you can define the following types of triggers:

Insert	Code that fires whenever a new record is added.
Update	Code that fires whenever a record is changed in a specified table.
Delete	Code that fires whenever a record is deleted from a table.

To use triggers, write the code that you want to execute whenever these events occur. Below is an example of an update trigger function that validates the state field against a list of possible states:

```
*--- This function validates the state field.
FUNCTION Customer_update_trigger
LOCAL ll_retval
ll_retval = .t.

*** --- validate data
IF NOT INLIST(customer.state, "WA", "CA", "OR")
    ll_retval = .f.
ENDIF

RETURN ll_retval
```

If the state is valid, the function allows the changes to be made. Otherwise the function returns false, which causes Visual FoxPro to return the data to its previous state.

The following code results in the modification of a history bucket value whenever a field value is updated:

```
*--- This function updates a bucket amount stored
*--- in a ytd_sales table.
FUNCTION Invoices_update_trigger
LPARAMETERS pcCustno, pnUpdAmt

SELECT YTD_SALES
IF SEEK(pcCustno)
    REPLACE YTD_SALES.TOTALSALES WITH ;
            YTD_SALES.TOTALSALES + pnUpdAmt
ENDIF
RETURN
```

Table Properties

Validation Rule: ...

Validation Text: ...

INSERT Trigger: ...

UPDATE Trigger: ...

DELETE Trigger: ...

Comment:

OK **Cancel**

Using Data Buffering

Along with a powerful data dictionary, Visual FoxPro includes a feature that greatly simplifies the creation and editing of data. This feature is known as data buffering. Data buffering takes the place of the old SCATTER/GATHER techniques used in FoxPro 2.x. Where in FoxPro 2.x it was necessary to copy your data into memory and edit the values stored in memory, Visual FoxPro now handles this process for you automatically.

Enabling Buffering

Visual FoxPro provides a number of methods for enabling its buffering mechanisms. The first is a function called CURSORSETPROP. This function allows you to specify different parameters that Visual FoxPro uses when accessing table data; one of these parameters is called **Buffering**.

The following code shows how to turn buffering on for a table called customer:

```
SET MULTILOCKS ON
=CURSORSETPROP("Buffering",2,"Customer")
```

Notice the use of the SET MULTILOCKS command. Because buffering maintains multiple locks, you need to turn on Visual FoxPro's ability to maintain multiple locks with the SET MULTILOCKS command.

Next, notice the call to the CURSORSETPROP function. This function is called using three parameters. The first parameter is the name of the property you want to change, in this example it is buffering. The second parameter is the value you wish to change the property to. In this example you entered 2, and that puts the table in "pessimistic row buffering" (explained later). The last parameter is the alias of the table or view we wish to change buffering for. This last parameter is optional; if it is not specified then the property change occurs on the currently active table.

The second method of changing a table's buffering mode is found in the Form Designer. Later in this seminar you learn how to change the various properties associated with a form. Included in these properties are a set of properties related to the maintenance of buffering modes.

Buffering Modes

Visual FoxPro provides five different data buffering modes. These modes are:

- FoxPro 2.x Style Buffering, Default FoxPro Locks
- Pessimistic Row Buffering
- Optimistic Row Buffering

- Pessimistic Table Buffering
- Optimistic Table Buffering

Each of these buffering modes controls the buffering and locking of buffered data. In order to understand what these different modes of buffering do, you need to understand the terms used to define this tool.

The terms pessimistic and optimistic refer to how Visual FoxPro locks data. The pessimistic method locks data as soon as a user begins editing a record, whereas an optimistic method locks data when the user decides to update the record. Pessimistic locking prevents two users from changing the same set of data, while optimistic locking allows this behavior.

Row buffering and table buffering are different methods of controlling the number of records that can be buffered at one time. Row buffering buffers a single record at a time, whereas table buffering allows you to buffer multiple rows at a time.

Now that you understand the terms used in buffering, you can see that Optimistic Table Buffering allows users to make changes to multiple records, and those records are locked as soon as the actual records are updated. Pessimistic Row Buffering buffers a single record and locks data as soon as editing begins.

Saving and Canceling Changes

Now that you understand how to buffer your data, you need to know how to save and cancel changes to records. Visual FoxPro provides two methods for controlling these functions. These methods are `TABLEUPDATE` and `TABLEREVERT`. `TABLEUPDATE` commits changes made to a record to the Visual FoxPro tables, while `TABLEREVERT` rolls those changes back to their original state.

Using the Form Design Tools Objectives

Some of the biggest changes in the Visual FoxPro development environment are found in the Form Design tools. Visual FoxPro has the look, feel, and control of a true Windows application, and with this new set of capabilities comes a new set of tools. This section discusses the use of these new tools.

In this chapter you:

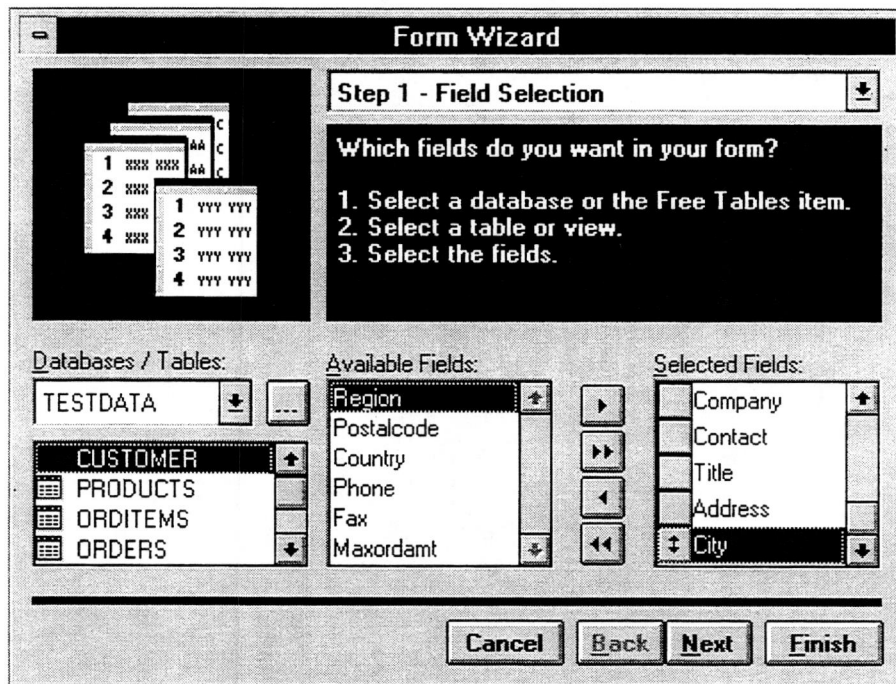
- Develop forms rapidly using the Form Wizard.
- Learn the components of the form designer.
- Use the new Form Designer Tools.
- Use the new Form Designer Controls.
- Use the FoxPro Grid Control.
- Add PageFrames (Tab Interfaces) to your forms.
- Use timer controls.

Using the Form Wizard

In today's high pressure development environments, it is nice to have a development tool that takes some of the pressure off. Visual FoxPro provides a tool that speeds the development of data entry forms: The Form Wizard.

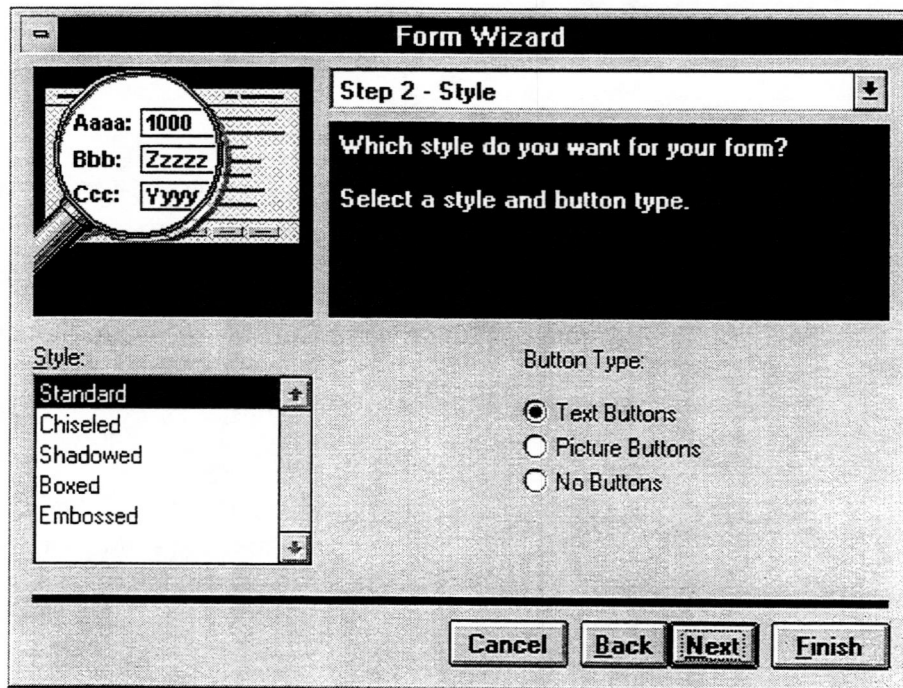
The FoxPro Form Wizard is a user friendly interface that creates data entry forms rapidly and efficiently. The Form Wizard is capable of creating two types of forms, *single table forms* and *one-to-many forms*. This section discusses the creation of single table forms; one-to-many forms are discussed later.

To create a new form, select the Documents-Forms section of the project outline and click New... You are then given the option of either using a wizard or creating a form from scratch. Choose wizard to activate the Form Wizard, and the following screen is displayed:



This screen is the Table and Field Selection screen of the Form Wizard. To create a new form with the wizard, select the table you wish to create a form for. If no tables are open, or if the table presented in the Tables list box is not the one you desire, click the button with the ellipses (...) to open a new table.

After you select a new table, you can select the fields to be added to the form by using the four buttons located between the two list boxes at the lower right of the form. These buttons move the contents selected from the available fields list to the selected fields list. Now you can move on to the next screen:



This screen allows you to specify style parameters for your form. The first parameter is the general style of the form. You can choose from five different styles:

- Standard
- Chiseled
- Shadowed
- Boxed
- Embossed

You can also specify the style of buttons to be used for the form. The wizard provides the choice of picture style buttons, text style buttons, or no buttons at all. Upon selecting these styles for the form, you can now move on to specifying a sort order for the form.

Form Wizard

Step 3 - Sort Order

How do you want to sort your records?

Records will be sorted according to the order of the selected fields. You may select up to three fields.

Available fields

- Cust_id
- Company
- Contact
- Title
- Address

Selected fields

- City

Add > **< Remove**

☒ Ascending ☐ Descending

Cancel **Back** **Next** **Finish**

By default, the records in an opened table appear in natural order. The *Sort Order* screen allows you to specify the order in which records are displayed. Use the list box to select up to 3 fields to sort your records. This process adds a new index to the FoxPro table you are creating this form for.

WARNING! If you specify sorting options, Visual FoxPro will build indexes on the fields you select for sorting. If this file has a large number of records in it, this process could take a long time.

After you specify a sort order, the wizard displays the finishing screen that gives you a few more design options. You can specify a title that will be displayed on your form. You can also choose from the following methods of exiting the wizard:

- Save the form for later.
- Save and run the form.
- Save and modify the form in the Form Designer.

The last option is discussed in the next section.

The screenshot shows a window titled "CUSTOMER" with a standard Windows-style title bar. Inside the window, there are several text input fields with labels to their left: "Cust_id:" with the value "DRACD", "Company:" with "Drachenblut Delikatessen", "Contact:" with "Sven Ottlieb", "Title:" with "Order Administrator", "Address:" with "Walserweg 21" and a vertical scroll bar to its right, and "City:" with "Aachen". At the bottom of the window, there is a row of ten buttons: "Top", "Prev", "Next", "Bottom", "Find", "Print", "Add", "Edit", "Delete", and "Exit". The "Next" button is highlighted with a dashed border.

The finished screen shown above was generated with text buttons and a standard appearance interface.

If you want to change the entire appearance of your form, you can go through the wizard again and change any of the form generation options you originally chose. If you want to look at the form before the final stages, select Preview from the finishing screen of the Form Wizard. The screen below was generated using the Embossed form style and picture style buttons.

The screenshot shows a Visual FoxPro form titled "CUSTOMER". The form has a title bar with the word "CUSTOMER" and standard window controls. Below the title bar, the word "CUSTOMER" is displayed in a large, bold font. The form contains several data entry fields with labels to their left:

- Cust_id: ALFKI
- Company: Alfreds Futterkiste
- Contact: Maria Anders
- Title: Sales Representative
- Address: Obere Str. 57 (with a vertical scroll bar)
- City: Berlin
- Region: (empty)
- Postalcode: 12209
- Country: Germany
- Phone: 030-0074321
- Fax: 030-0076545
- Maxordamt: \$6,300.00

At the bottom of the form, there is a horizontal toolbar with several icons: a left arrow, a right arrow, a double left arrow, a double right arrow, a printer icon, a document icon, a list icon, and a folder icon.

As you can see the Form Wizard creates very useful and powerful forms with just a few mouse clicks. The Form Wizards use a set of templates stored in a file called WIZSTYLE. This file is a Visual FoxPro class library that you can extend and modify thus changing the forms generated by the Wizards. But, before you can begin exploring these templates, you need to know how to modify your forms with Visual FoxPro's design tools. That is what is coming next...

Designing Forms with the Form Designer

Now that you know the value of the Form Wizards, you can better understand what these wizards do, as well as how you can create forms without the wizards.

The form wizards are limited to creating only the most simple of forms. In an application that needs to go beyond simple forms, you need to know how to generate them yourself.

This section covers the Form Designer tool in detail. The different interface tools are discussed along with how to successfully use forms in your own applications.

The Form Designer

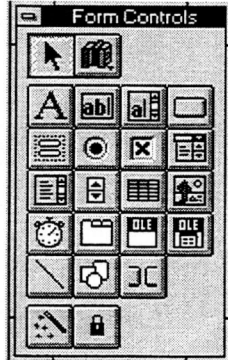
The first step in creating a new form is to activate the form designer from the Project Manager. After doing so, you can begin the creation of your form.

Before learning to actually generate new forms, explore the tools found in the form designer.

The form designer provides many tools to assist in the development of forms:

- The Form Controls Toolbar
- The Property Sheet
- The Layout Toolbar
- The Data Environment
- The Color Palette Toolbar

The Form Controls Toolbar

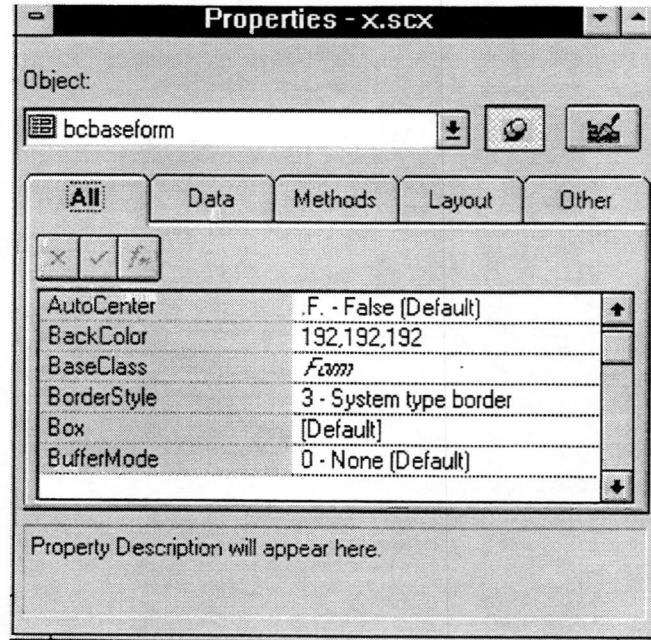


The first item you are going to explore is the form controls toolbar. The form controls toolbar allows you to select the objects that make up your form. The controls that are available in Visual FoxPro are:

- Text Boxes
- Combo Boxes
- Grids
- Push Buttons
- PageFrames (Tabbed interfaces)
- Spinners
- Images
- Text Labels
- Radio Buttons
- Custom controls
- OLE Objects

and others.

The Property Sheet



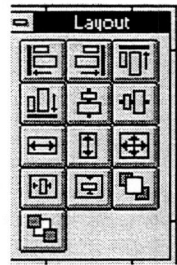
Each control placed on a form has a set of properties associated with it that define its characteristics. These characteristics may include:

- Font
- Size
- Color
- Caption
- Control Name
- Control Data Link

and other control specific properties.

You can change these properties by updating them in the properties sheet.

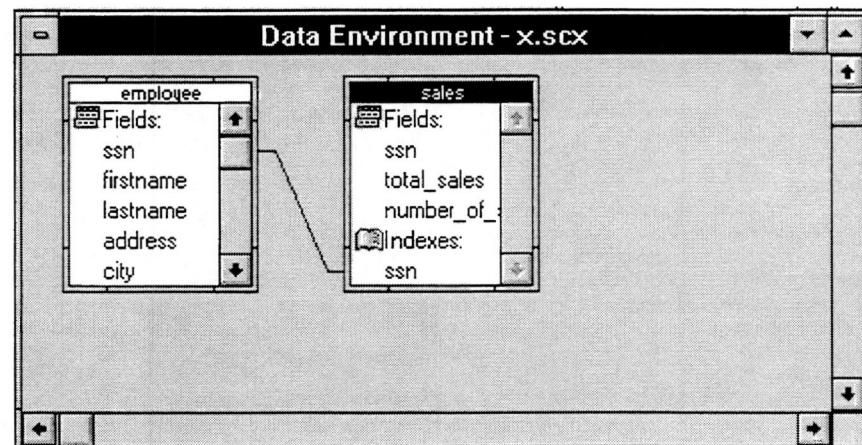
The Layout Toolbar



The layout toolbar allows you to change the visual appearance of controls on a form. Some of the features found on the layout toolbar include:

- Setting tab order
- Setting alignment of controls
- Specifying size of controls

The Data Environment



FoxPro 2.x provided an option when you created forms to save something called the data environment. In FoxPro 2.x it was not wise to save this environment since it sometimes caused unforeseen problems.

Visual FoxPro includes a radically enhanced data environment tool. The *Data Environment* allows developers to specify which tables are used for a form by placing them in the Data Environment. You can activate the data environment by clicking the right-mouse button and selecting Data Environment from the activated popup.

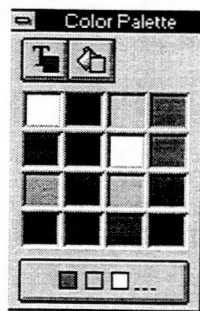
The Data Environment is very similar to the data dictionary tool. When you right click on the Data Environment, a popup similar to the one in the data dictionary appears that allows you to add tables.

Specifying the Data Environment

The first step in designing your form is to specify its data environment. To specify a form's data environment, right-click on the form designer surface and select Data Environment from the resulting popup.

After selecting Data Environment from the popup, you can add tables to the data environment of the form. Note that the form Data Environment works exactly like the Data Environment in the Report Designer. Review the Report Designer section to better understand this concept.

The Color Palette Toolbar



The last tool of interest is the color palette toolbar. This allows you to select from a set of predefined colors when changing the colors of an object.

Adding Controls to Your Forms

Before you add controls to your forms, review the Form Designer Terms that you are going to encounter when developing your forms:

Form Design Terms	
Controls/Objects	Controls and objects are the building blocks for your forms. Examples of objects include: Forms, Buttons, Pictures, Check boxes, and OLE controls.
Properties	Properties are the attributes of an object, like its colors and font. Visual FoxPro have a very complete list of properties that describe the look and feel of all your objects. Examples of properties are colors, typeface, height, and width.
Events	Events are the actions that your objects need to respond to. In many Xbase languages there were generally two events you trapped for, a When event and a Valid event. These events also exist in Visual FoxPro, but they are now accompanied by a list of dozens of new events.
Methods	Methods are actions you can invoke against objects from within your programs. Methods are special procedures/functions attached to an object. These methods are similar to the snippets you coded for special tasks when dealing with screens and screen objects.

The next step is to add a control to your form. The easiest way to add a new control to your form is to drag and drop a field from one of the tables in the forms data environment. This places a control on the form and sets its appropriate data properties.

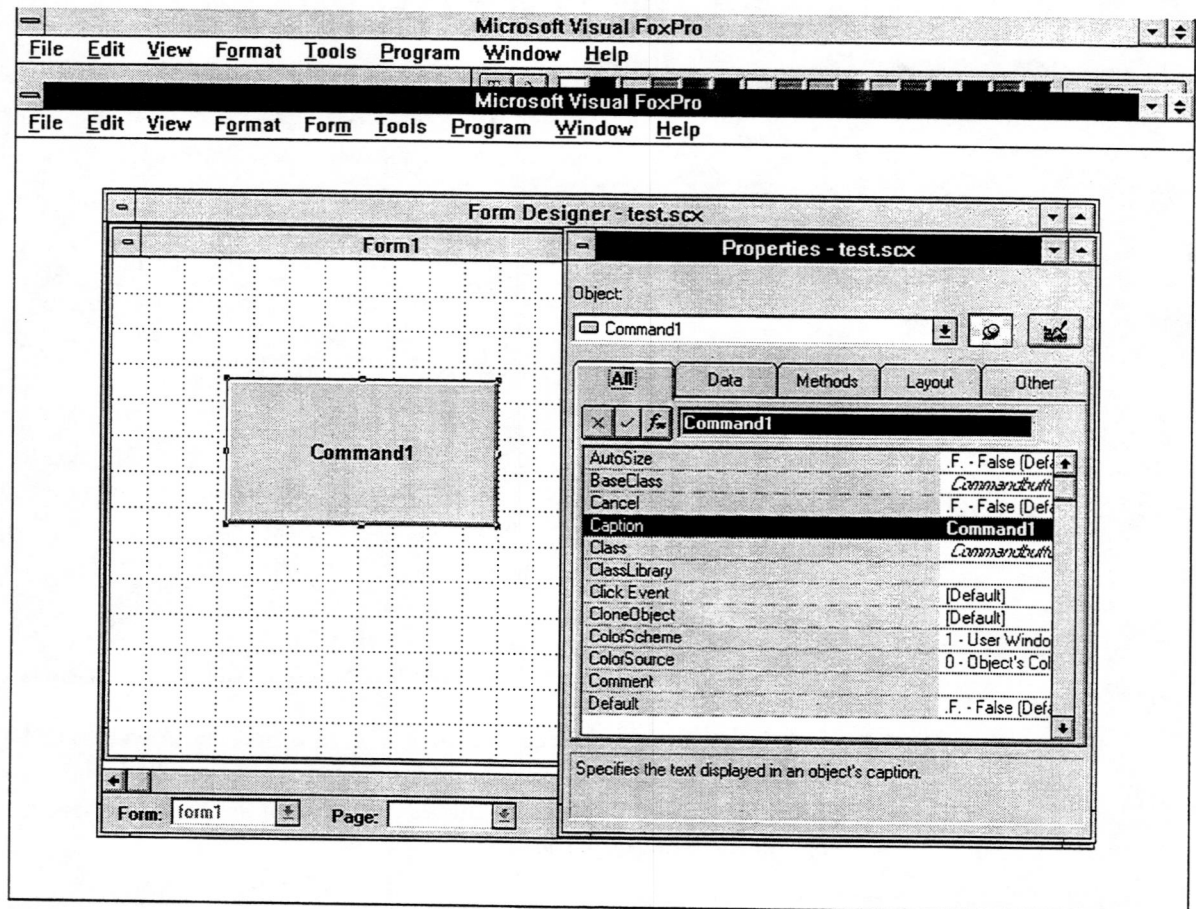
Another useful feature of the data environment is that you can drag an entire related table from the data environment and FoxPro automatically establishes a relationship with the appropriate parent table.

The more common way to add a control to your form is to select the appropriate control from the controls tool palette and highlight where this control is placed. After placing a control on your form, you can set the appropriate properties for the newly added object.

Setting Properties

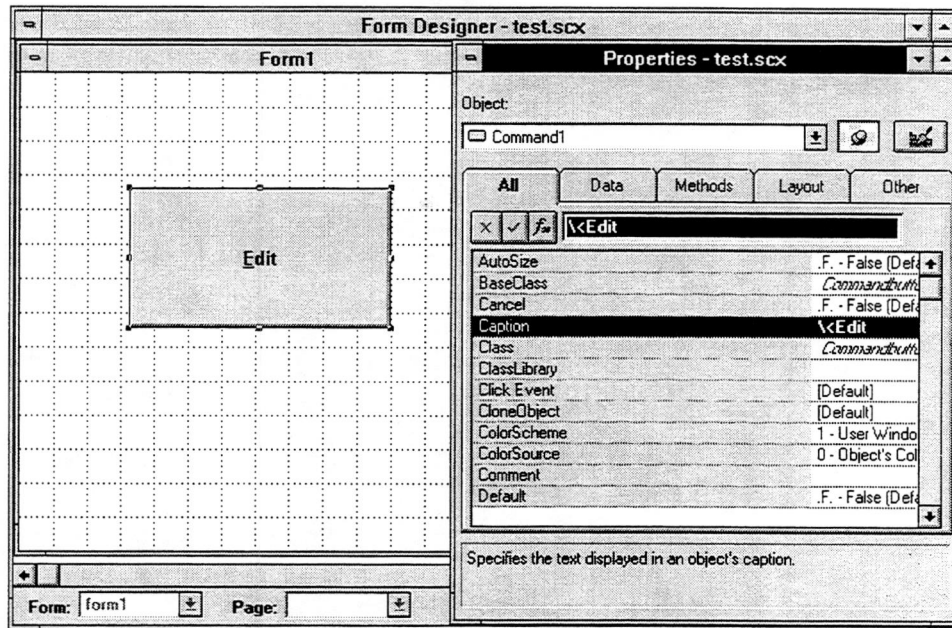
After adding a control to your screen, you can change the properties of this control. As you may recall, you can change the properties of a screen object by using the Properties Sheet. You can access the property sheet in two ways. You can select the Properties option from the View menu or you can select the object and Right-Click with your mouse.

After activating the Properties Sheet, you can begin changing the properties of the item you selected.

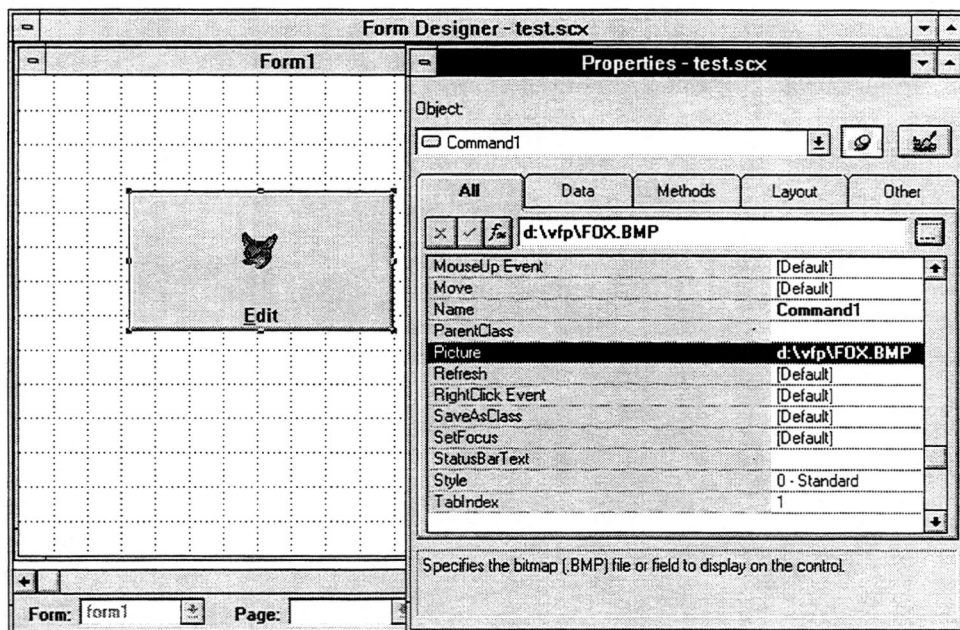


Most objects have some kind of caption associated with them. The above figure shows the property sheet with the caption selected. You can change the caption by selecting the caption property on the properties sheet and changing it to a caption you want.

If you want a button to be named Edit, with E being a hot key equivalent, enter \<Edit as the caption property. See the example below:

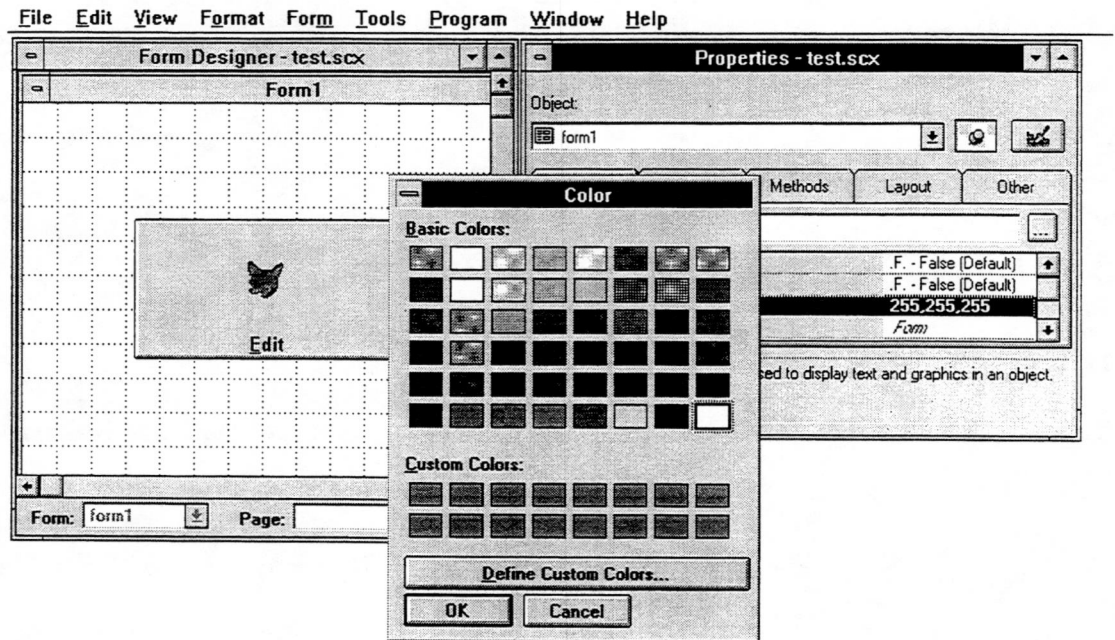


Another feature of the new form builder is its ability to combine a bitmap and text on the same button. All you need to do is change the picture property of your form to your custom icon. The example below illustrates how to change the bit map for a push button:



FoxPro 2.6 has an object menu option that allows you to alter the appearance of most objects; Visual FoxPro has this same function.

The screen below demonstrates how to change the color of a selected item. Select the BackColor property of the object, click the ellipses (...) next to the property value, and select a color from the palette. As you can see, you have full control of the selected colors.

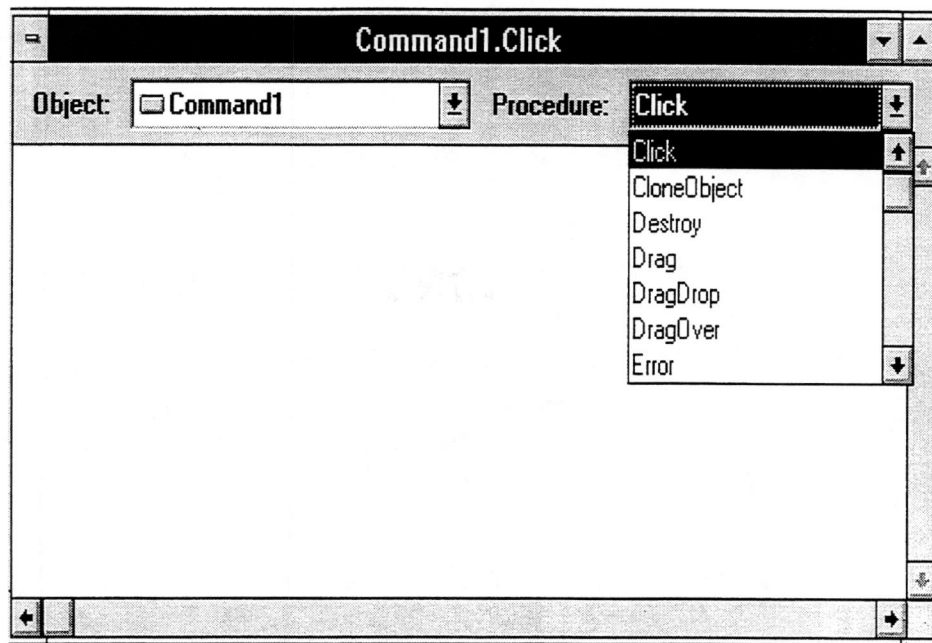


Understanding Events

Visual FoxPro is a true event-driven Windows application. Each object associated with a form has a set of Windows events that it can respond to. Examples of Windows events are:

- A mouse click
- A key press
- The movement of a window
- The activation of a window.

These are just a few examples. In Visual FoxPro you can attach code to any object or event. To attach code to an object, activate the appropriate controls event editor by double-clicking on that object. This brings up a dialog similar to the one below:



Notice the drop-down list on the right side of this figure. These are a few of the events that are available for a push button display. For now, you are going to add some code to the click event of your button. Type the following code into the event editor:

```
WAIT WINDOW "BUTTON PRESSED"
```

Now run the form by clicking the exclamation mark (!) on the standard toolbar. When you click the button on your form, FoxPro executes the above code.

Another event of interest is the VALID event. This event is left over from FoxPro 2.x; it allows you to validate whether a field contains valid data before you attempt to exit it. Place the function you developed previously in the valid event of a text object. Insert the following code to test the validity of the state field.

```
IF g_valstat(employee.state)
    llRetVal = .t.
ELSE
    WAIT WINDOW "Invalid State Entered"
    llRetVal = .f.
ENDIF
RETURN llRetVal
```

If you enter an invalid state, this warning is displayed.

Using Methods In Your Code

In the definitions section you learned that events are actions your users can invoke against an object. Note that you can also invoke your own events from within your code, with code known as *methods*.

Methods are pre-defined actions you can execute against an object. You can execute methods from your code using a coding concept known as the *Object . Property / Object . Method* syntax. This syntax is consistent with the syntax used in Microsoft Visual Basic and Visual Basic for Applications (VBA).

In earlier versions of FoxPro, you addressed screen objects using the names of their GET fields. Because of Visual FoxPro's object-oriented nature you can now deal with objects directly using the *Object . Property / Object . Method* syntax.. Visual FoxPro provides a set of special system variables that you can use to address your objects from within your programs. These system variables are:

- `_screen`
- `ActiveControl`
- `ActiveForm`
- `ActiveFormSet`
- `Parent`
- `THIS`
- `THISFORM`
- `THISFORMSET`

In Visual FoxPro you need to be very specific in addressing the objects that you want to change. In FoxPro 2.x you only dealt with a single read at a time. In Visual FoxPro you can have dozens of forms active at a time, with no one form being specifically active or in control. The code below demonstrates how to change the properties of a control in a generic manner:

```
*-- Function ChangeColor
*-- This code changes the background color of an object
FUNCTION ChangeColor
  _screen.ActiveForm.BackColor = RGB(255,255,0)
RETURN
```

```
*-- Function RestoreColor
*-- This code changes the background color of an object
FUNCTION RestoreColor
    _screen.ActiveForm.BackColor = RGB(255,0,255)
RETURN
```

The above commands use the Object . Property syntax. The object addressed is `_screen.activeform`. The property changed is the `BackColor` property. Object Methods uses syntax similar to the Object Property syntax although `Object.Method()` is used here. Notice the addition of parentheses. Remember that methods are simply object functions.

The Refresh Method

In FoxPro 2.x, you used the `SHOW GETS` clause to refresh your screens and objects. Visual FoxPro has replaced this with a method called *refresh*.

Whenever you programmatically change the `Value` property of a screen text object, FoxPro does not automatically refresh the screen. As a developer, you have to instruct FoxPro to refresh the screen or screen object you changed.

The code below illustrates two methods of refreshing screen objects:

The first

```
THISFORM.edtCustomerLastName.Value = "Paddock"
THISFORM.edtCustomerLastName.refresh()
```

changes the value of `edtCustomerLastName` and refreshes the object to reflect the change.

The second

```
THISFORM.edtCustomerLastName.Value = "Paddock"
THISFORM.refresh()
```

changes the value of `edtCustomerLastName` and refreshes the entire screen to reflect the change.

Other objects have methods that require additional parameters. If you add a combo box to your screen, you can provide the data source in a number of different ways:

- You can use a Table.
- You can use an SQL statement.
- You can add the list items manually with the AddItem() methods.

The code below demonstrates how to load a combo box with the AddItem method:

```
*-- Initialize combo box with west coast cities  
THISFORM.combo1.AddItem("San Francisco")  
THISFORM.combo1.AddItem("Los Angeles")  
THISFORM.combo1.AddItem("Portland")  
THISFORM.combo1.AddItem("Seattle")  
THISFORM.combo1.AddItem("Vancouver")
```



The screen above shows the results of the AddItem clause executed on a form.

The Grid Control

The *Grid Control* is one of the most requested and eagerly awaited features of Visual FoxPro. A grid is a spreadsheet-like view of multiple records in one or more tables. The default appearance of a grid is almost identical to the Xbase

command, BROWSE. Although, grids in Visual FoxPro are much more flexible and useful.

In some Xbase languages, it was very difficult to integrate a BROWSE with other objects on a screen, but it is easy to integrate a grid in Visual FoxPro. *With grids, you can easily create a window showing data in two tables that are linked in a one-to-many relationship.*

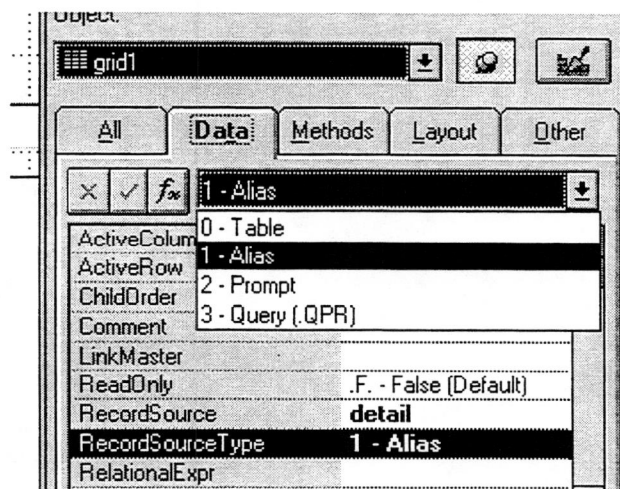
Like forms, formsets, and page frames, grids are *container* objects. Every component of a grid, including the *cells* that display data, is an individually controllable object. You can control every aspect of the behavior and appearance of each component individually. In fact, fields displayed in a grid do not have to be text boxes; they can be check boxes, combo boxes, or any other control that is reasonable for the underlying data.

When you add a grid object to your forms, you need to set a few properties that control the behavior of the grid. The following are some of the behaviors that can be controlled:

- Size and color of the grid
- The source of data for the grid
- The columns displayed in the grid

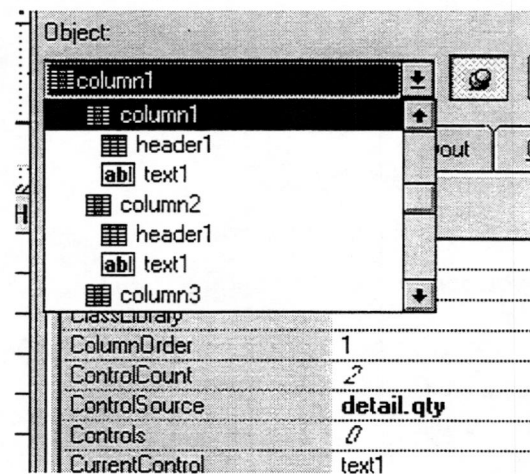
At a minimum you need to specify the RecordSource and RecordSourceType properties in your grid control. The RecordSource property determines what table or view the grid uses to fill its lists. The RecordSourceType property specifies what data container type the RecordSource is. The following can be the RecordSource of a grid:

- An open alias
- A table within a database
- A local view
- A free table



By default, FoxPro displays all columns contained in a table defined by the RecordSource. In most situations, you only want to specify the columns that are displayed. This will enhance your application's performance because it limits the amount of data that must be loaded prior to displaying the grid. You need to manipulate two properties to accomplish this, the ColumnCount and the ControlSource.

The default ColumnCount value set by FoxPro is -1. The ColumnCount property controls how many columns are displayed in the grid, and -1 displays all columns. Upon changing this property, FoxPro allows you to change the properties of each column individually. The figure below illustrates a grid with the ColumnCount property set. Now all you need to do is change the ControlSource for each column to the column name(s) of your table.



Other properties you can change for a column include:

- Column header
- Column color
- Column size

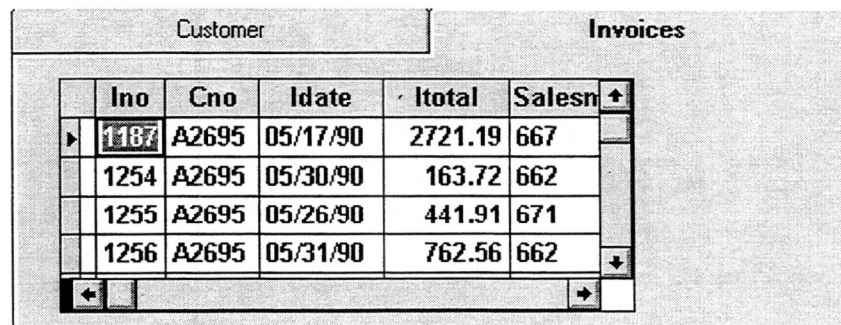
Using PageFrames (Tabbed Interfaces)

A rule of thumb you learned earlier in this course limits the number of objects on a form to groupings of no more than seven to nine items. There are times when, despite your best efforts, a form has too many items. Even if you are willing to put many objects on a form, there may not be enough room for them.

In the past, this problem was typically solved with a multiple page form, or with a set of forms (In FoxPro 2.x, this was called a Screen Set). You can produce complex data entry screens this way in Visual FoxPro using a Formset. However, Microsoft has also provided another alternative that it has pioneered over the past couple of years in its other products, known as a *tabbed interface*, or more properly, a *Page Frame*.

Though the Page Frame is a form that can have multiple pages, it no longer uses the awkward mechanism of having a numbered series of pages (1 of 4, 2 of 4), perhaps with push buttons to advance forward or backward through the screens. The Page Frame looks like a well-organized file drawer of manila folders. Each page has its own tab projecting from the top and is labeled with a meaningful name. To move between pages in any desired sequence, simply click on the appropriate tab, and that page comes forward.

Page Frames provide an intuitive interface for organizing information related to a single entity. Visual FoxPro's Options dialog is an example of such an interface. The figure below shows a form with two Page Frames; one frame shows customer information and the other one shows the customer's invoices.

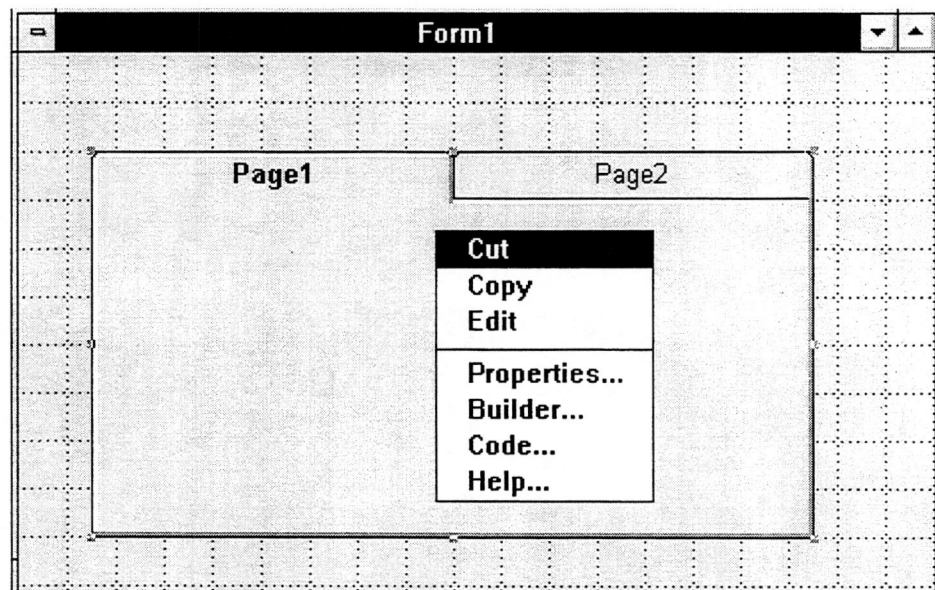


Ino	Cno	Idate	Itotal	Salesn
1187	A2695	05/17/90	2721.19	667
1254	A2695	05/30/90	163.72	662
1255	A2695	05/26/90	441.91	671
1256	A2695	05/31/90	762.56	662

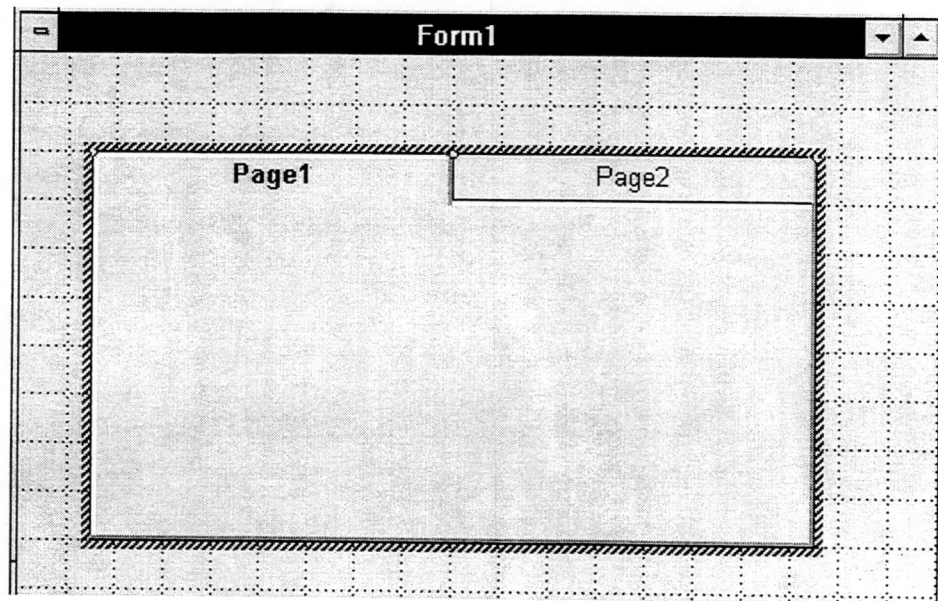
Following are some of the most common properties you modify when using the Page Frame:

- *PageCount*: The number of tabs on the Page Frame.
- *TabStretch*: Controls whether the tabbed dialog can stack tabs rather than reduce the size of each tab.
- *Caption*: Each tab has a property that displays text on that tab.

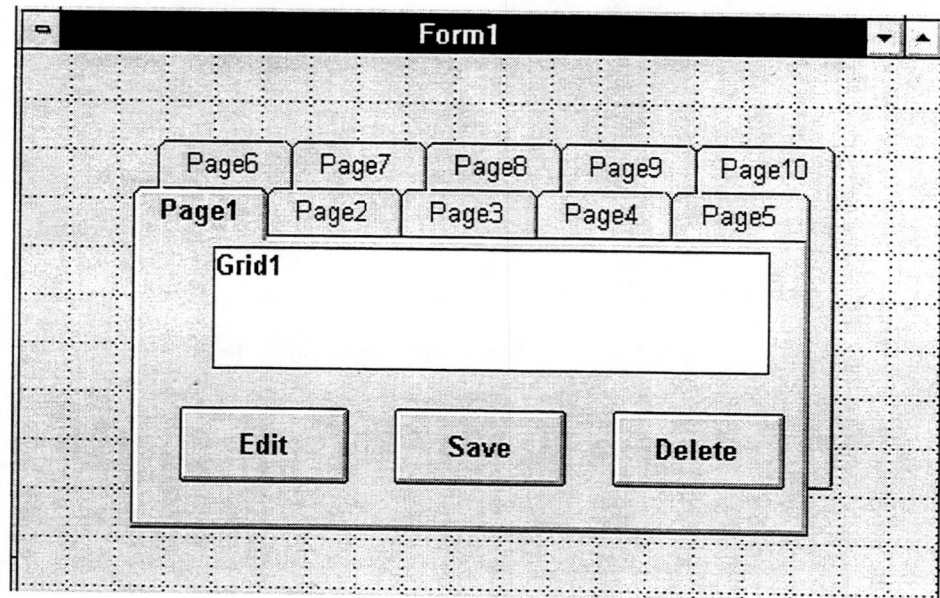
After adding a Page Frame to your form, you can add controls to each page on the Page Frame. To add controls to a page, you need to activate the Page Frame for editing. To activate a Page Frame, right-click on the object and select edit from the resulting popup.



After you select Edit from the popup, FoxPro highlights the Page Frame with a wide border, as shown below. You can now select the appropriate page and add controls from the tool palette.



The following illustration shows a Page Frame with the PageCount property set to 10, the tab stretch property set to stacked, and with individual captions for each page.



The Timer Control

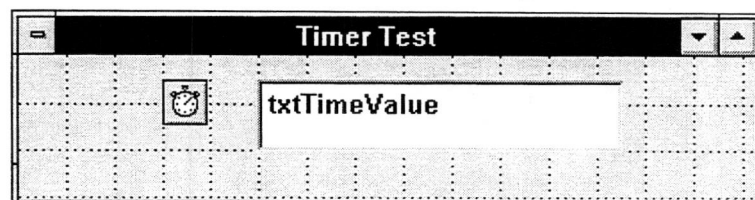
The *Timer Control* allows you to tell FoxPro to execute a program or procedure at a specified interval. These procedures may include checking a network directory for the existence of specific files, closing records that have been locked for a long interval, and polling serial ports for data at a specific time.

When you use a timer control, you need to concern yourself with the interval property and the timer event. The *interval property* specifies how often the *timer event* fires. The interval property is measured in milliseconds, so if you want to fire an event every second, you need to set the timer property to 1,000. Whenever the interval is reached the timer event fires, and this is where you specify the code to run.

The following code illustrates a form that looks in a directory for the existence of a file in a network directory. If the file exists, it is opened and processed automatically.

```
*-- Polling code to add imported files to master
*-- tables.
ln_rows =ADIR(la_files, "C:\IMPORT\*.DBF")
IF ln_rows > 0
  FOR ln_kount = 1 TO ALEN(la_files,1)
    USE (la_files[ln_kount,1]) IN 0 ALIAS a_tmpf1
    ln_record = RECCOUNT()
    SCAN WHILE NOT EOF()
      *-- Code to add file to master tables here
    ENDSCAN
    *-- Close and delete file
    USE IN a_tmpf1
    ERASE(la_files[ln_kount,1])
  ENDFOR
ENDIF
```

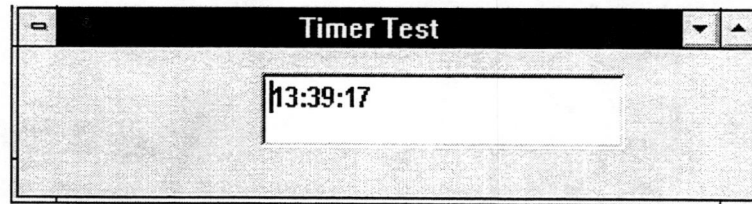
The following illustration shows a form with a timer control and a text box at design time:



The following code demonstrates how to create a digital clock using a timer and a text control object:

```
*-- Display time in text object  
THISFORM.txtTimeValue.Value = TIME()
```

Now look at the form at runtime:



NOTE: The timer control is not shown on the form at runtime. Timer controls are only visible at design time.

Creating Information From Your Data Objectives

In this section you:

- Query your data using Views
- Access Client/Server data using views
- Learn the definition of parameterized views and how to use them
- Format your output using the Visual FoxPro reporting tools
- Combine the power of Views and the Visual FoxPro Report Designer

Querying Your Data Using Views

One of the most powerful aspects of Visual FoxPro is its ability to query large sets of data efficiently. Visual FoxPro uses an index optimizing technique known as Rushmore to access large sets of data in a rapid manner and takes this ability one step further with the inclusion of a tool known as a view.

In the simplest of terms, views are SQL queries that behave exactly like tables. In FoxPro 2.x it was common to create reports by selecting a set of data, placing it in a cursor or temporary file, and then processing that file through the report writer to create your output. The following listing shows how a report was commonly created using FoxPro 2.x:

```
SELECT customer.lastname, customer.firstname, ;
       orders.order_no, orders.order_id;
FROM customer, orders ;
WHERE customer.cust_id = orders.order_id ;
ORDER BY lastname, firstname ;
INTO TABLE tempfile
```

```
REPORT FORM CUSTLIST TO PRINT
```

This was the common approach to creating reports in FoxPro 2.x. Visual FoxPro now has the ability to take the SQL SELECT statement and store it in a database with a name. By doing this, Visual FoxPro allows you to create reports rapidly with a couple of minimal commands. The report above can now be generated with the following syntax:

```
USE customer_orders_view
REPORT FORM CUSTLIST TO PRINT
```

This is just a brief review of the power of views. Now it is time for you to explore the views in more depth.

Understanding View Types

There are two types of views that can be used in your Visual FoxPro applications. They are Local and Remote Views. Local views allow you to store predefined queries against Local (.DBF) style data. Remote views allow you to access data from client/server (remote) databases. This section will discuss the creation and use of both remote and local views.

You can also update views by creating views that can update the sources of their data. This section provides the information necessary for you to create views, both read only and updatable.

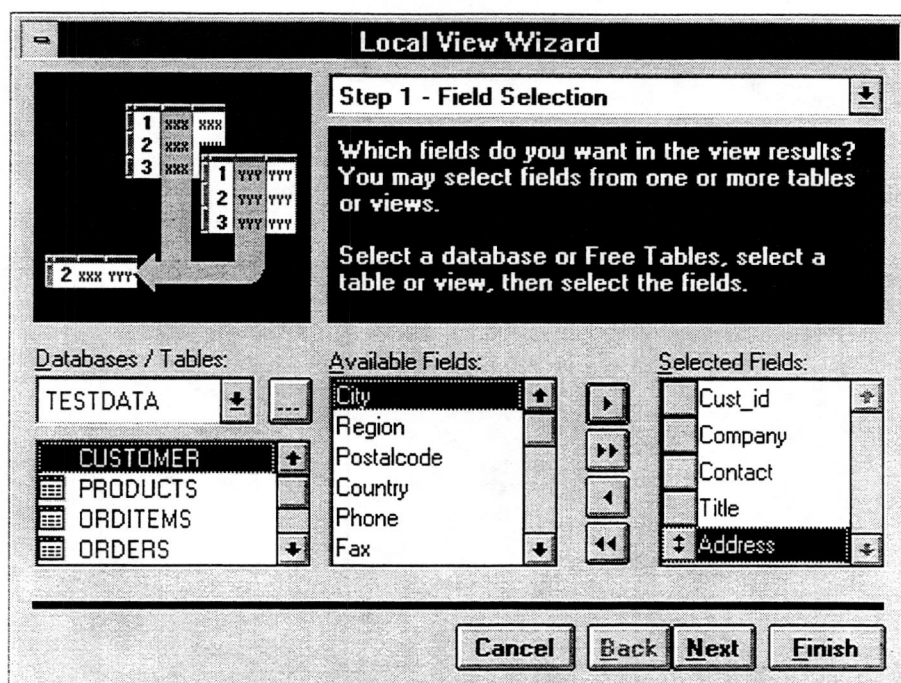
NOTE: In order to use views in your application you need to use a database container (Visual FoxPro Data Dictionary). Visual FoxPro stores all view management information in the database container.

Using the View Wizard

The easiest way to begin using views is with the View Wizard. The View Wizard is a tool that is useful for selecting the columns, sorting, and filtering criteria for the data that will be displayed in your view. The View Wizard like all of the other Visual FoxPro wizards provides a simple set of selection screens that you can use to create a view

To use the View Wizard, first select the Database-Local Views option from the Project Manager and then click New. FoxPro now gives you the option of either creating a view by hand or using the View Wizard. If you select the View Wizard you are prompted with a series of screens that provide the selection criteria for your view.

The first screen provided by the View Wizard is the Field selection screen. This screen allows you to select the fields from the table(s) you wish to query with your view. The Field Selection screen is shown below:



Use this dialog to select the table(s) you wish to query. If you have no tables open, click the ellipsis (...) button to bring up the *Table Open* dialog. Select a table; FoxPro fills the list of available fields with a list of fields from the currently selected table. From the list of available fields, you can select the fields you wish to have in your view.

After you have selected the fields for your view, click the Next button and move to the *Sort Order* criteria screen. This screen allows you to order your view by a single field or combination of fields. If you wish to order your view by city, move the city field from the available fields list to the selected fields list. The following *Sort Order* dialog illustrates this point:

Local View Wizard

Step 3 - Sort Order

How do you want to sort your records?

Records will be sorted according to the order of the selected fields. You may select up to three fields.

Available Fields:

CUSTOMER.Cust_id
CUSTOMER.Company
CUSTOMER.Contact
CUSTOMER.Title
CUSTOMER.Address

Selected Fields:

CUSTOMER.City

☒ Ascending
☐ Descending

When querying large amounts of data, you need to filter the quantity of data returned by your view. The Query Wizard provides a *Filtering* screen that allows you to limit the set of data presented in your view. The following screen demonstrates the view Filtering screen:

Local View Wizard

Step 4 - Filtering

If you want to specify certain records for your local view, create an expression using the Field, Operator, and Value boxes.

Click Preview to see the results.

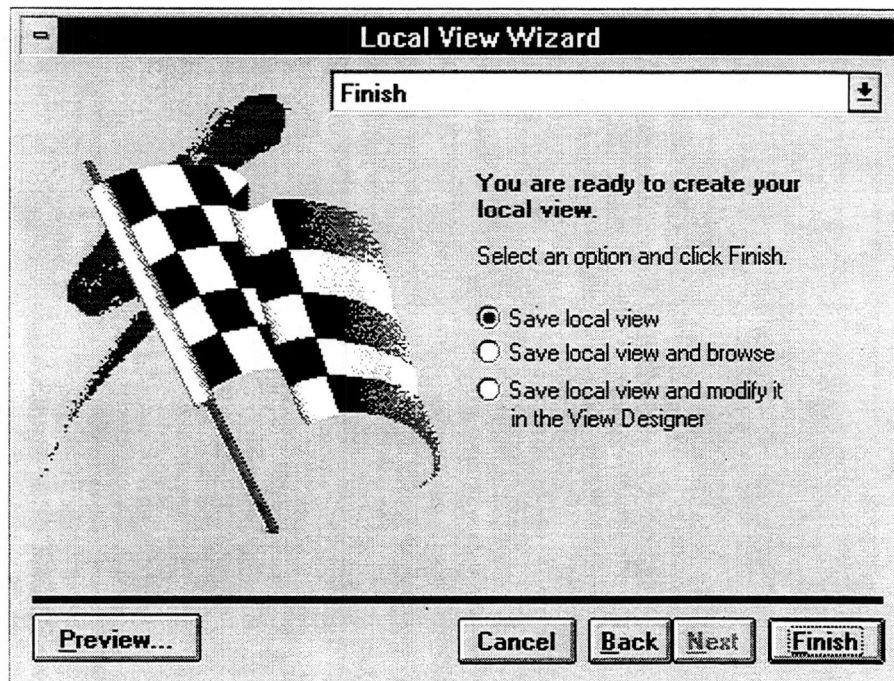
Field: CUSTOMER.CUST_ID Operator: equals Value:

☒ And
☐ Or

Field: CUSTOMER.CUST_ID Operator: equals Value:

The filtering screen allows you to specify query filter criteria and gives you the ability to include one or two fields in your filter criteria. You have the

option of creating an AND or an OR style filter condition. A query of [Give me all the people in the state of "WA" AND who have salaries over 50,000] represents an AND condition. [Give me all the people in the state of "WA" OR who have salaries over 50,000], represents an OR condition. Only two fields are available for your filter conditions.



When you complete the filter criteria for your view, exit the View Wizard by one of the following methods:

- Save the view.
- Save and browse the view.
- Save the query and modify it in the View Designer.

After you create queries using the View Wizard, you can modifying them in the View Designer. When developing an application it is common to begin your queries with the View Wizard, and then modify them with the View Designer. This allows you to overcome some of the limitations of the View Wizard while still benefiting from its ease of use.

Starting the View Designer

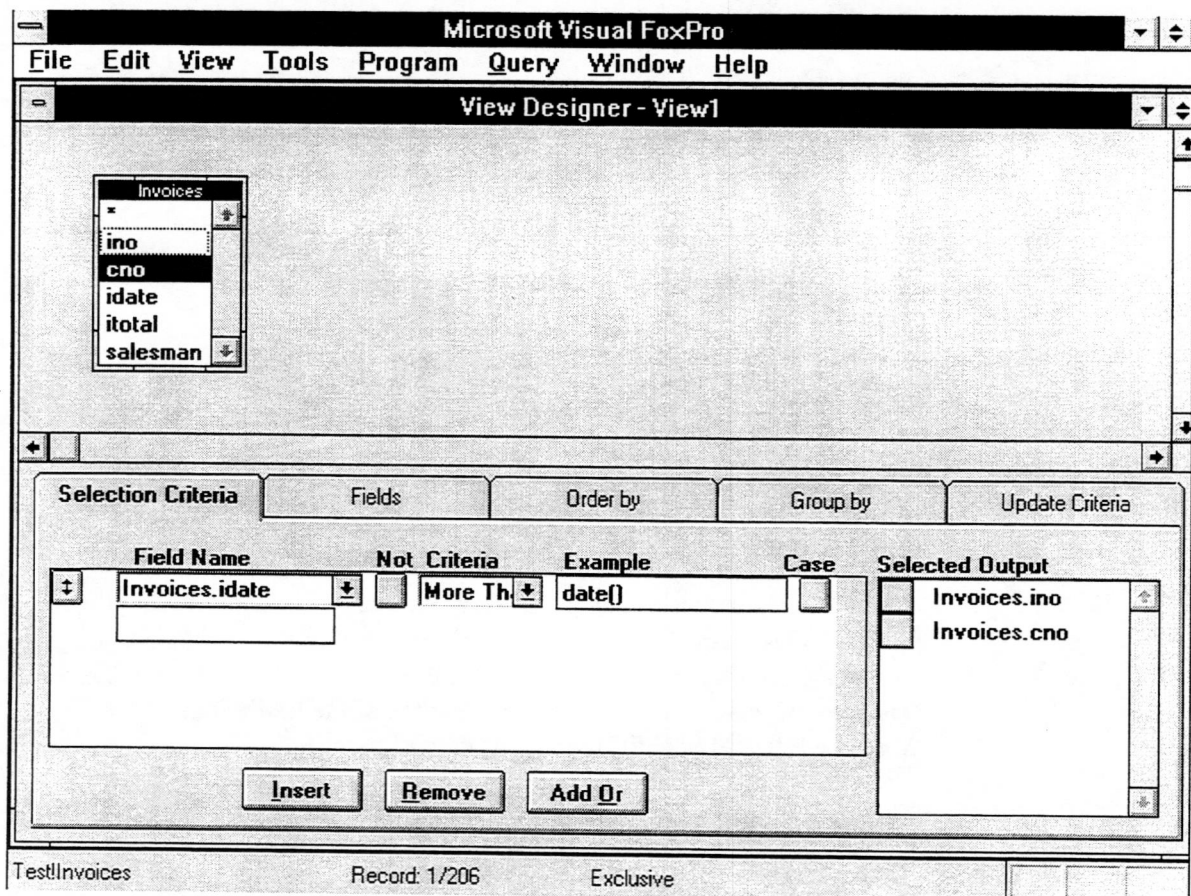
You can modify your views a number of ways:

1. You can modify your view by opening your database from the command window and typing MODIFY VIEW <your view name>. This will open your view in the View Designer.
2. Open the database container using the MODIFY DATABASE command and select the view from the database Designer.
3. Select the view from the Project Manager and press the Modify button.

Any method you choose launches the View Designer with your view ready for modification. The next section explains how to use the View Designer to extend the views created by the View Wizard.

Using the View Designer

When you select a view to modify you see a dialog similar to the one below. This is the View Designer; it is from this dialog that you specify the criteria that make up your view.



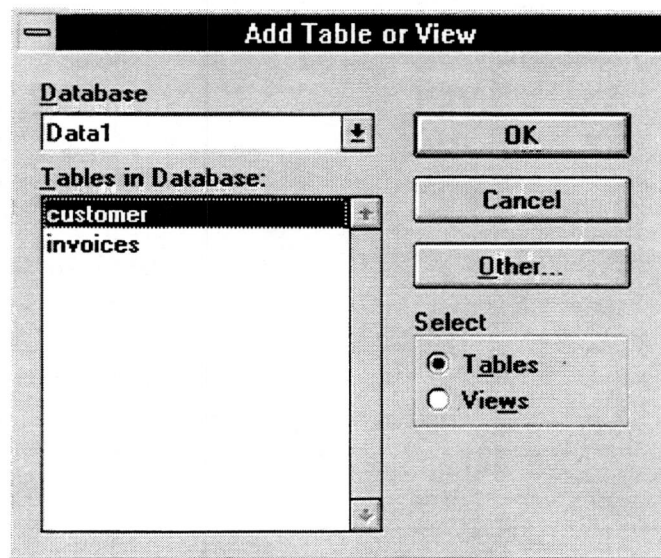
To define a view:

1. Select the tables for your view.

2. Specify the fields you want to be contained in your view.
3. Specify selection/filter criteria for your view (optional).
4. Specify the sorting and grouping of the data for your view (optional).
5. Specify the criteria for Visual FoxPro to use when updating data contained in your view.

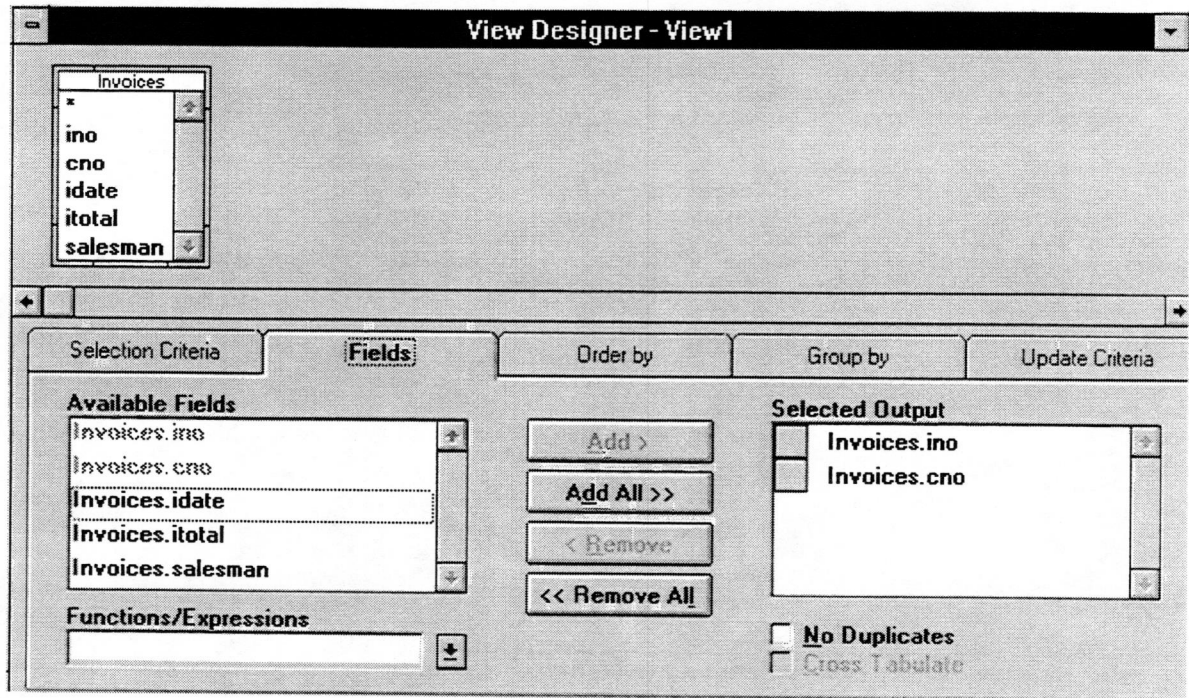
Step 1: Select Tables For Your View

The first step in defining a view is to select the tables that you want to use to form the basis of your view file. When you create a new view, the Add Table or View dialog prompts you to select the table or tables that you want to create your view for.



Step 2: Specify Fields For Your View

Select the fields you want your view to contain. Either double-click the field in the field list box, or select the fields tab from the View Designer. To select all fields, double-click on the (*) found in the table field list. Visual FoxPro inserts the fields for your view into the Selected Options field list box found in the lower right section of the View Designer. The illustration below demonstrates selecting fields for a view.



Steps 3 and 4: Specify Selection, Sorting, and Grouping Criteria.

You can filter the records to be shown in a view by using *Selection Criteria*. This Selection Criteria is essentially the WHERE clause of a SQL SELECT statement. The illustration below demonstrates specifying selection criteria for a view.

View Designer - View1

Invoices

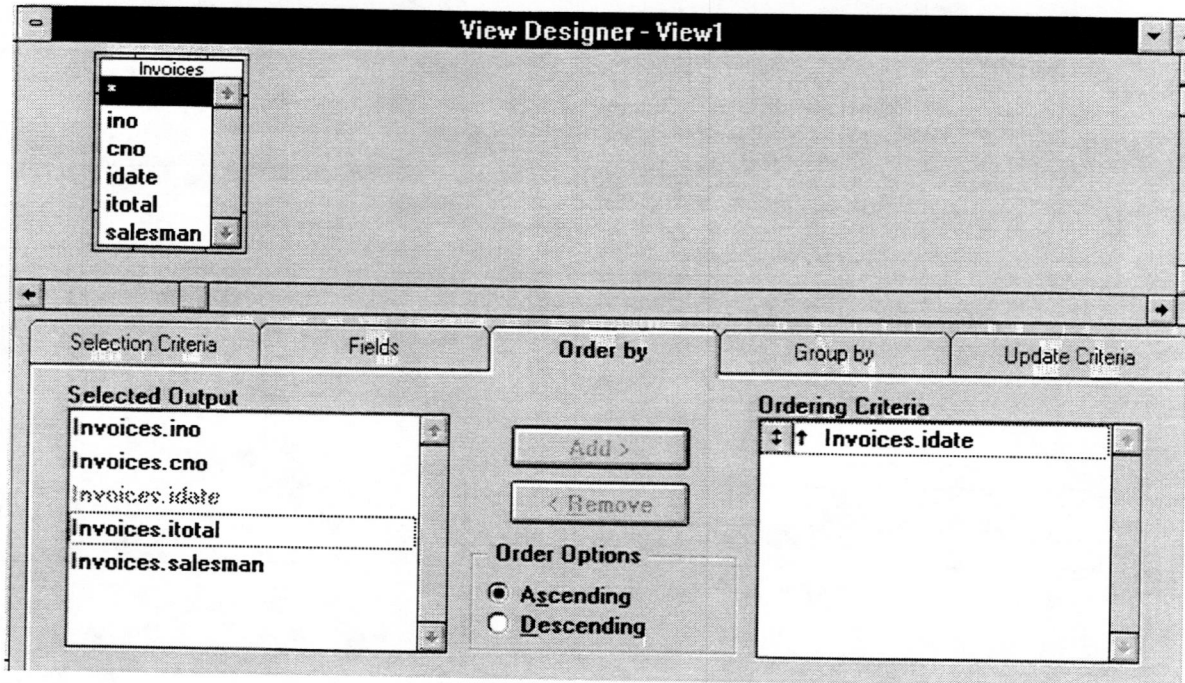
- ino
- cno
- idate
- itotal
- salesman

Selection Criteria | **Fields** | **Order by** | **Group by** | **Update Criteria**

Field Name	Not	Criteria	Example	Case	Selected Output
Invoices.idate	<input type="checkbox"/>	More Than	date()	<input type="checkbox"/>	Invoices.ino Invoices.cno

Insert **Remove** **Add Or**

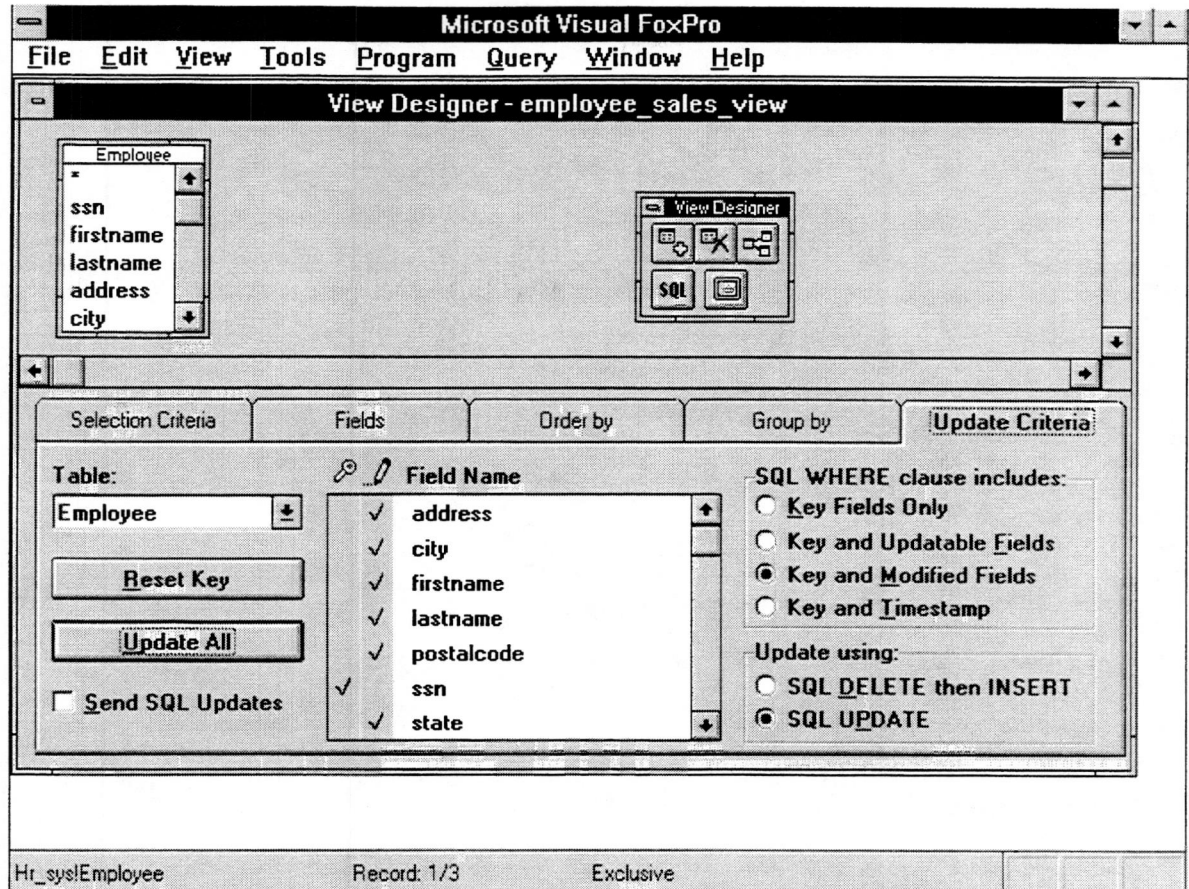
You can also specify sorting and grouping criteria for your views. The illustration below demonstrates setting view sorting criteria.



NOTE: If you choose to use grouping in your view, you cannot update the records found in the view.

Step 5: Specify the Update Criteria

The last step in creating a view is to specify its update criteria. You already know that you can create views that are updateable or read-only. If you choose to create an updateable view, you need to specify which fields are updateable and how Visual FoxPro updates these fields. The illustration below demonstrates specifying update criteria for a view.



The *Update Criteria* tab in the View Designer controls the method that Visual FoxPro uses to update the fields in the view. The first item of interest is the list of Field Names. Visual FoxPro requires that a key be specified for the entire list of records in the view.

Visual FoxPro uses SQL to update the data in a view. You need to identify a unique key for each record in order for SQL to find the correct record. To specify a key, click the button under the key icon of the View Designer. After you specify a key, you can then check the fields to be updated in your query.

The last option to check is the Send SQL Updates option. This, with the combination of other options, makes your view updateable. To summarize the steps necessary to create an updateable view:

- Select the fields for your view.
- Specify optional sorting and filter criteria.
- In the Update Criteria tab, specify the key field(s) to be updateable.
- Check the Send SQL Updates option.

Now you can open the view from the command window by typing the following commands:

```
OPEN DATABASE HR_SYS
USE employee_view
BROWSE
```

The real advantage of using views is that they encapsulate information about your database at a high level (in the database container). From this encapsulation, you can change the following old 2.x code:

from

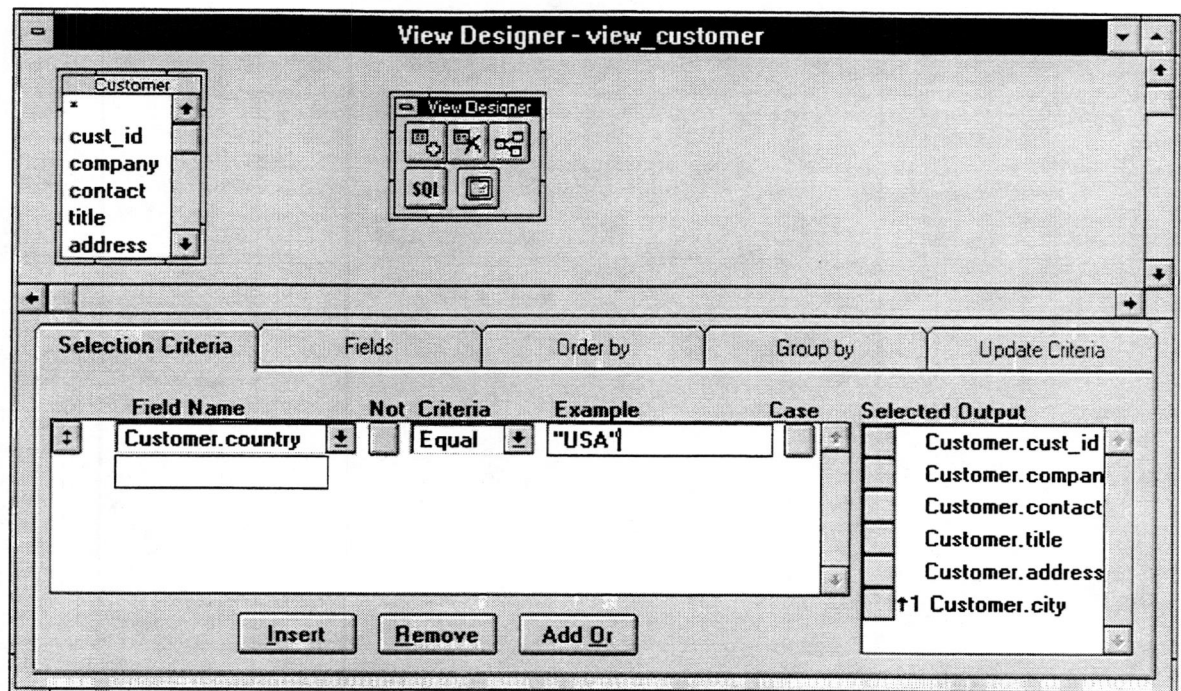
```
SELECT *
FROM Customer ;
WHERE customer.state = "WA" ;
INTO CURSOR c_sales
REPORT FORM salesrep TO PRINT
```

to

```
USE sales_view
REPORT FORM salesrep TO PRINT
```

Creating Parameterized Views

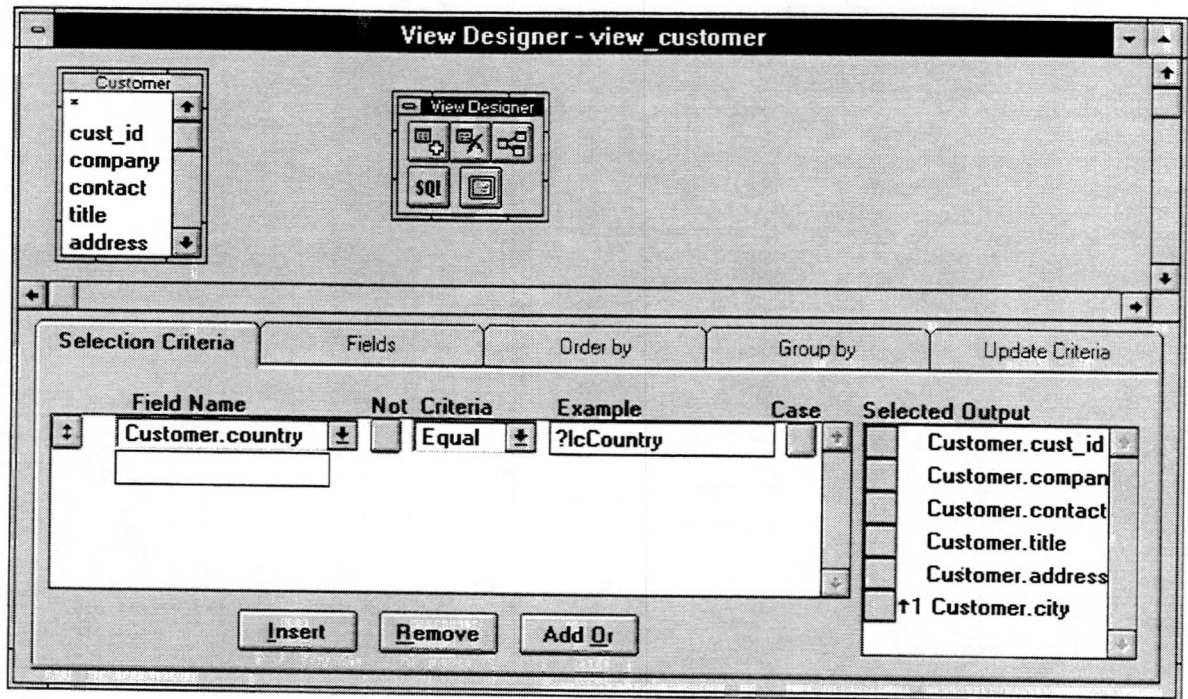
In the example above, you learned how to select data from a local table based on a static set of pre-defined criteria. One item that makes Visual FoxPro's views very powerful is that you can specify ad-hoc parameters to drive your views. For example, what if you wanted to view customer records for a specific country, say the USA. One way of doing this is to select the Select Criteria tab from the View Designer and specify **customer.country = "USA"** as a selection criteria (see below).



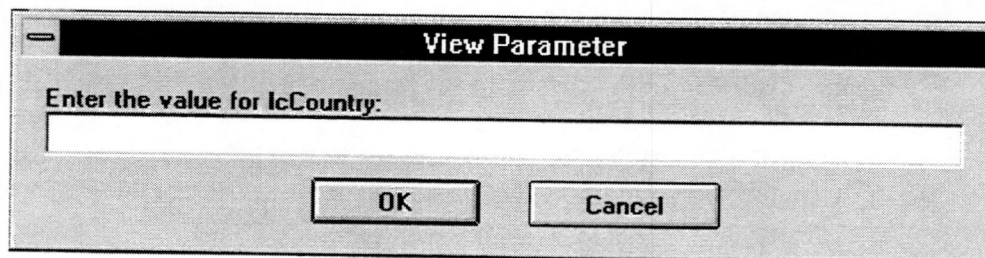
However, this method is rather limited because the selection criteria is set to the static value of "USA". What if you wanted to do an ad-hoc system where the value of the country changes from time to time? How would you accomplish this? You can do this with a mechanism known as a parameterized view. All you have to do is change the Selection Criteria to:

customer.country = ?lcCountry. The question mark (?) specifies lcCountry as a parameter that Visual FoxPro looks for in memory when the view is opened. If Visual FoxPro finds a variable named lcCountry in memory, it substitutes the value stored in lcCountry for the selection criteria and runs the view. The following figure shows the View Designer with a parameter called lcCountry.

WARNING! When you specify the name of your parameters, be careful not to specify the same name as a field in your table, as this can cause unforeseen problems.



If Visual FoxPro does not find the variable named lcCountry, it prompts the user to enter the value with a dialog similar to the following:



The ability to specify ad-hoc parameters in a view is convenient for creating small filtered data sets quickly. While this is nice in a local environment (just .DBFs), it is a necessity in a client/server environment. In a client/server environment you want to process sets of records in the smallest available increment; this is where parameterized views come in handy.

Querying Client/Server Information with Remote Views

As mentioned previously, you can access data from client/server systems using what is called a remote view. Remote views use Open Database Connectivity (ODBC) to access the data found on remote servers. With this capability, you can use data found in SQL Server, Oracle, Informix, and even your favorite mainframe, in your Visual FoxPro applications. The only difference is how you define the view.

Accessing data from client/server databases is a two step process in Visual FoxPro. Because FoxPro is engineered to use ODBC for accessing data, you first need to configure ODBC to access your remote server. This step requires you to configure something known as an ODBC data source. ODBC data sources allow you to specify the parameters used to connect to a remote server. These parameters include: the server driver (SQL Server, Oracle, Informix, etc.), the name of the database, and any server-specific connection parameters.

Upon defining an ODBC data source, you need to create a remote view. Remote views use ODBC data source information to connect to a server and allow you to create updateable views, much like you can do for a local view.

This section explains the advantages of using a client/server environment and shows how to connect to a remote data source.

What is Client/Server?

To begin a discussion of client/server, it is necessary to discuss what client/server is and what some of the probable pitfalls are.

Client/server is a communication mechanism where requests for data or commands are made by a client application. These requests are interpreted by some type of relational database management system (RDBMS), the server application, that returns a result set.

By their nature, database applications are network and resource intensive. Client/server allows you to shift some of the work of database applications from the network to the server, thus reducing the demand for network resources.

However, this shift in resources comes at a cost. Client/server applications are often more expensive and difficult to develop than normal PC type applications; Visual FoxPro reduces some of this difficulty.

Benefits of Client/Server

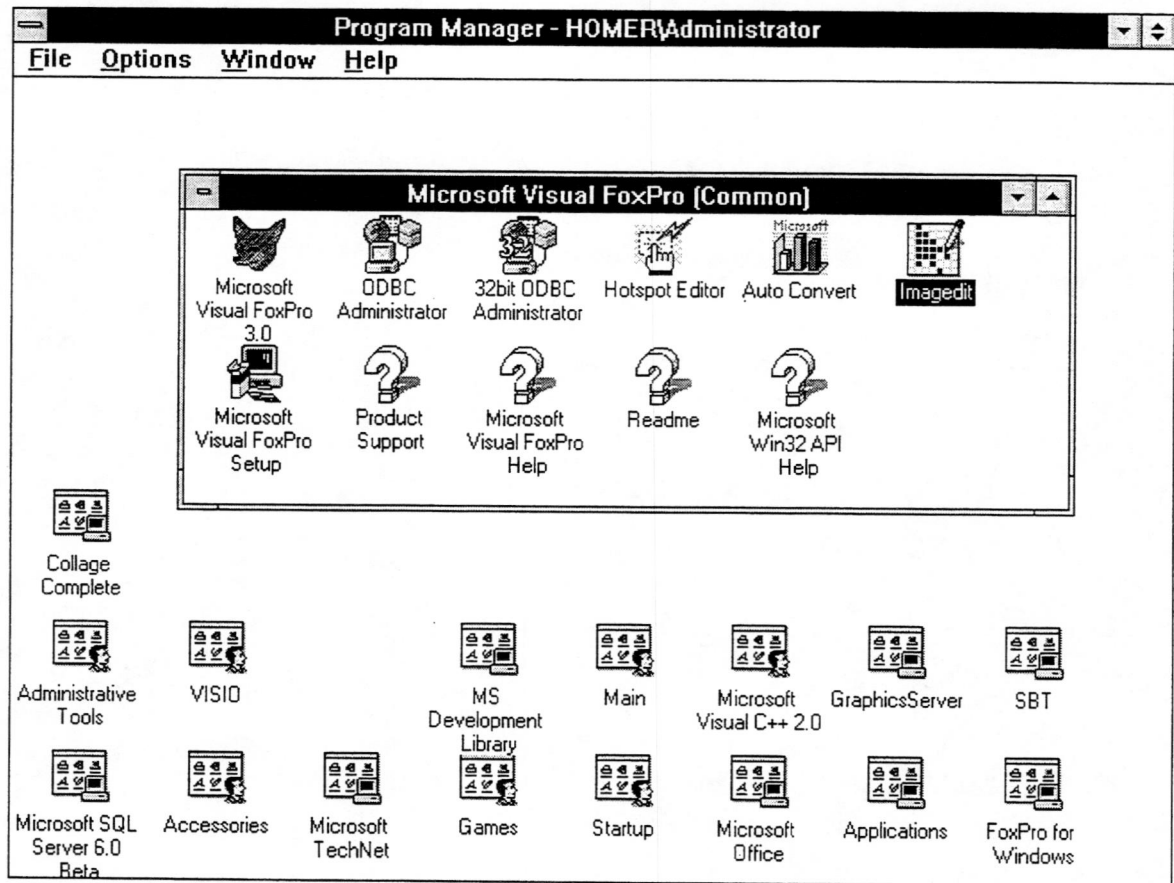
Client/server has more advantages than simply reducing resource and network utilization. Other benefits of using client/server include the following:

- *Access to Corporate Data.* Many companies store a great deal of data in legacy database applications. Client/server applications provide better mechanisms for accessing this legacy data.
- *Better Security.* Many PC databases (Visual FoxPro included) provide little or no support for data security. RDBMS applications are better at limiting access to data. Many databases can limit access to data at the row and column level.
- *Scalability.* One of the biggest advantages of client/server is its ability to scale up when resources become limited. Client/server allows companies to simply replace the back-end server with better, faster hardware. All of the client applications benefit from this upgrade. This is much cheaper than having to add new resources at the desktop.
- *Fault Tolerance.* Many of the RDBMS's on the market provide some level of fault tolerance. So, if your server goes down, your database remains intact with no corruption.

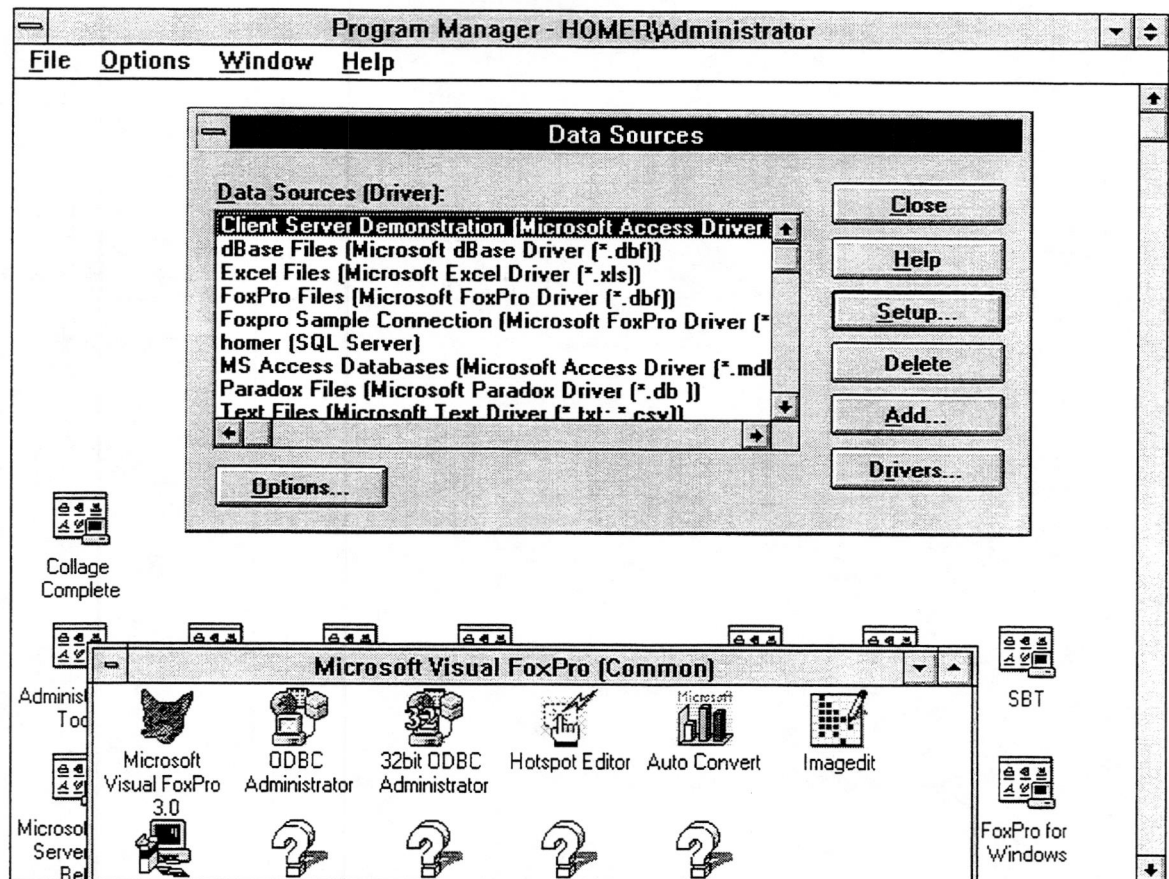
- *Distributed Data.* As companies grow, it is common for them to have data stored in different databases. These databases may be physically located at different locations. Client/server allows companies to store data in different locations and have this data appear as one logical database.

Accessing Client/Server Data from Visual FoxPro

Visual FoxPro uses ODBC as its mechanism for accessing server data. Before you can access data using ODBC, you must configure a data source. To configure an ODBC data source, activate the ODBC Administrator from the Program Manager, as shown in the following illustration:



When you activate the ODBC Administrator, the Data Sources dialog box seen below is displayed. This dialog box controls the configuration of ODBC data sources. From this dialog box, you can add new data sources or configure existing ones.



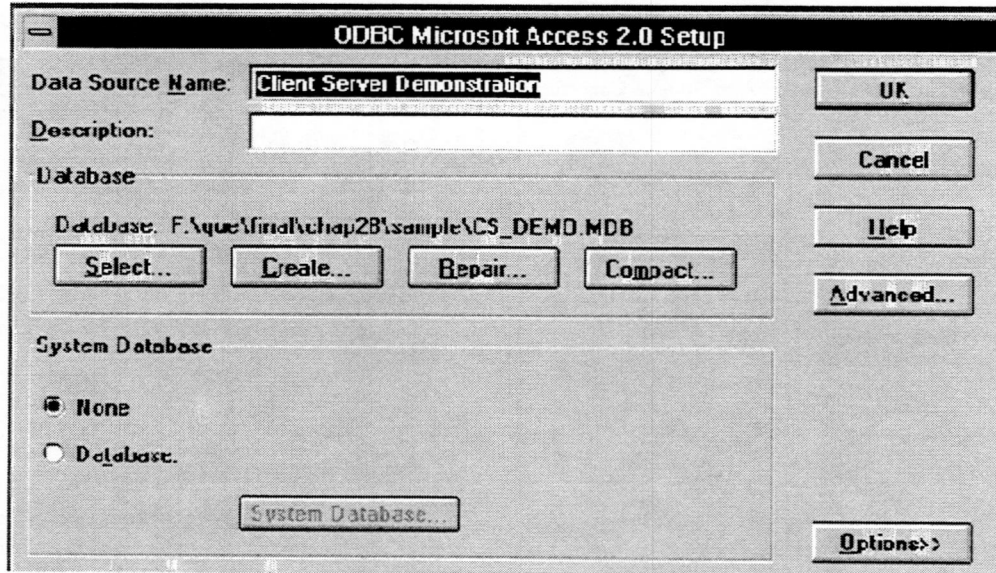
Click Add on the Data Sources dialog box. The ODBC Administrator then presents a list of its available drivers. These drivers, which are programs supplied by database vendors, create an ODBC link to their respective databases. Visual FoxPro ships with numerous ODBC drivers. These drivers include: FoxPro 2.0/2.5, MS-Excel, MS-Access, and SQL Server drivers.

TIP: If you want to experiment with client/server development, but do not have a server, look at the drivers provided with Visual FoxPro. Visual FoxPro provides FoxPro 2.0/2.5 drivers; you can use these to architect client/server applications immediately.

Configuring ODBC Drivers

When you select a driver, the ODBC Administrator opens the setup screen for the selected driver. The setup screen is responsible for the configuration parameters of an ODBC driver. This screen controls the specification of the data source name and the database connection parameters necessary for an ODBC conversation.

The following illustration shows the ODBC configuration screen for a Microsoft Access ODBC driver. The name of the data source is *Client Server Demonstration*. This driver requires that you specify the path of the Access database for a connection. Clicking Select activates the Windows Open File dialog box, allowing you to specify an available Access file.



Creating Remote Views

Visual FoxPro stores its ODBC access information within FoxPro database files. To establish a connection to a server, you must first create what is known as a *remote view*. Remote views are updateable queries that allow you to use server data as though it were FoxPro data.

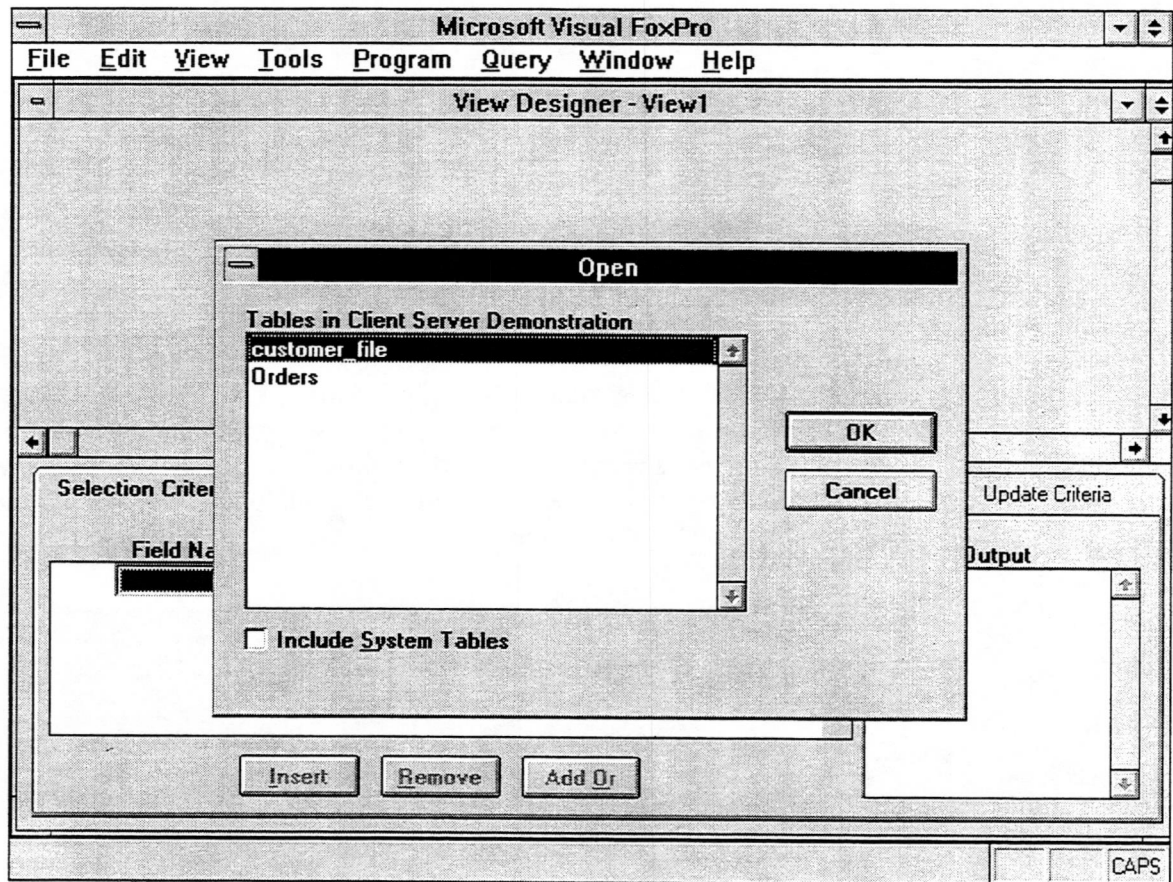
To create a remote view, click Remote View on the Database Designer toolbar. Next, connect to a remote source by using a *connection* or by *selecting from a data source*.

Connections allow you, the developer, to provide login and connection information to data sources. Selecting from the list of available data sources requires your users to provide login information to the server application each time the view is opened. Connections allow you to record and reuse login information for a specific data source. The following illustration shows the Connection Designer screen:

WARNING! If your remote system requires login id's and passwords, you can use connections to speed logging into remote servers. However, these login id's and passwords are stored unencrypted in the database container and are vulnerable to prying eyes.

When you select a data source or a connection, Visual FoxPro displays a list of tables contained in the remote database. The following illustration shows a

dialog screen that allows you to select the tables that provide the data for the remote view you are creating.



NOTE: The table selector dialog box provides an optional check box for selecting system tables. Most relational database systems provide a set of data dictionary files that specify information about the database. This information commonly includes table descriptions, column data types and descriptions, and index information. You can use this option to provide information to your users.

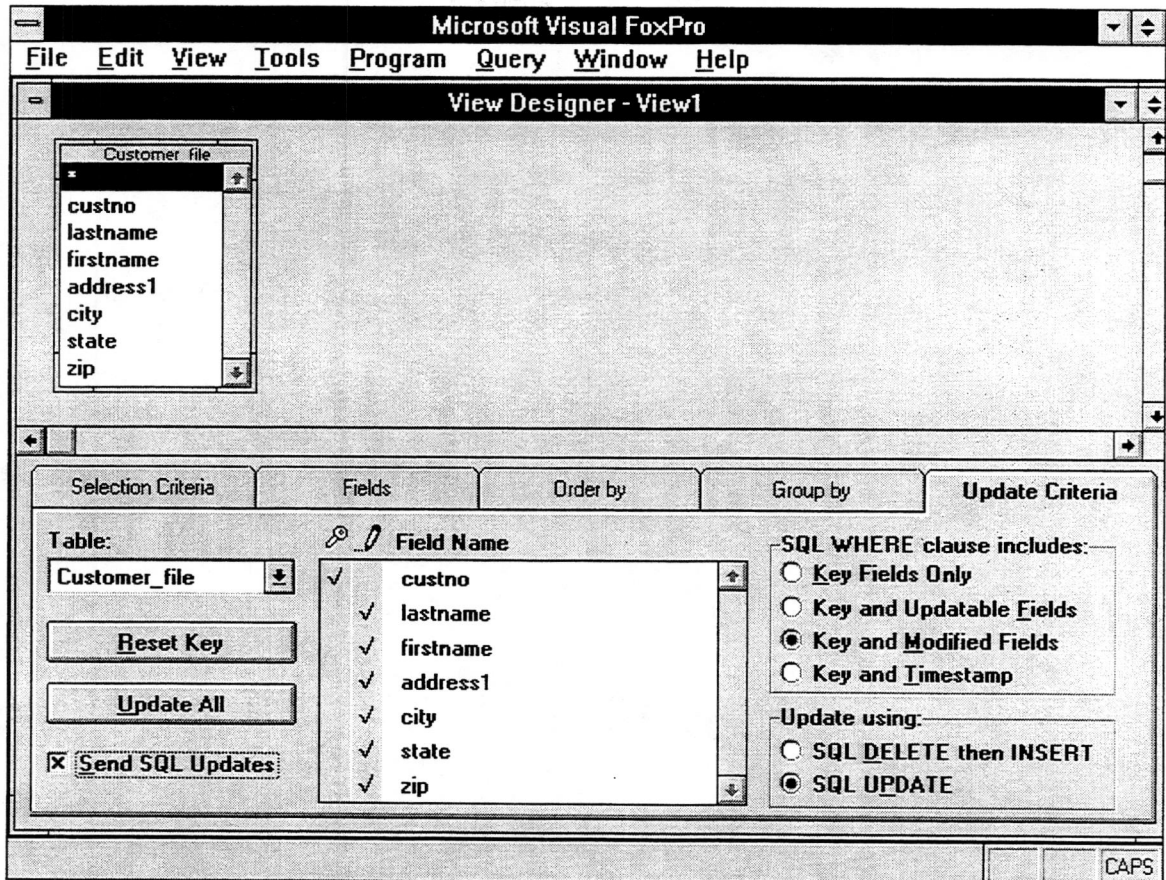
Once you select a table or set of tables for your view, Visual FoxPro activates the Visual FoxPro View Designer, shown in the following illustration.

The View Designer is the normal FoxPro Query Designer with one exception; this View Designer allows you to specify the parameters that define how data is updated.

The Update Criteria tab defines how Visual FoxPro manages the updating of data on the remote server. Visual FoxPro requires you to specify how and when data is updated on a remote server.

Each of the following update criteria is satisfied by a particular section of the update criteria page:

- Which columns are updateable
- The method for updating data on a server
- Criteria for establishing when data is updateable



The section that allows developers to specify which columns are updateable in a query is found under the Field Name heading on the update criteria page. To specify which columns are updateable, first provide a key for the file. This key can be a single field or a combination of fields that uniquely identify a record. Specify a key field. Then specify the fields that are eligible for update.

NOTE: After you select the fields for update, check the Send SQL Updates option. If this option is *not* specified, changes made to the remote data are *ignored*. Checking this option specifies that this is an updateable query.

Next specify how data is updated on the server. Visual FoxPro provides two methods for adding data to a server. You can use a SQL UPDATE statement, or a combination of SQL DELETE and SQL INSERT.

The main reason for choosing one method instead of the other is performance. Sometimes it is faster to delete and re-add the record than it is to update a column or set of columns. This depends on your particular database configuration.

The update success criteria is the last item to configure. When you work with remote database systems, take into account the fact that data may be updated by another user; this is called optimistic locking.

To make sure your tables are updated correctly, Visual FoxPro provides a set of parameters for checking whether remote data has been updated. Under the SQL WHERE clause header are the following four comparison criteria that determine when an update is successful:

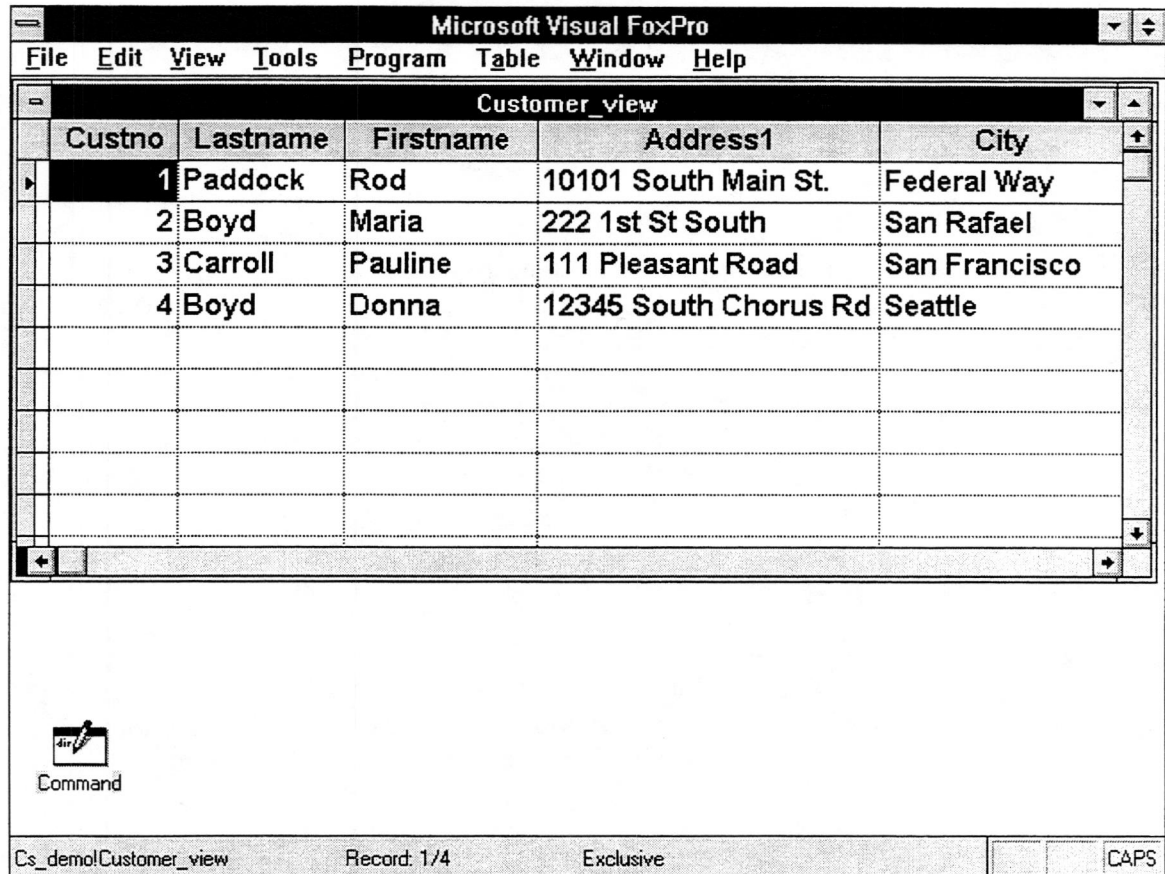
- *Key Fields Only*. Specifies that only the key fields are compared.
- *Key and Updateable Fields*. Specifies that only the key and updateable fields are compared.
- *Key and Modified Fields*. Compares the key and any fields that have been updated.
- *Key and Timestamp*. Checks the key and a time stamp for comparison.

If any of these criteria fails, the TABLEUPDATE() function returns false. This requires you to either abort the update or write a program that allows for the resolution of conflicts between client and server applications.

Using Remote Views in Visual FoxPro

You can use remote views in your applications as though they were FoxPro tables. The code below illustrates how to access data from an MS-Access database using a remote view. The following illustration shows the contents of an Access table using the BROWSE command.

```
OPEN DATABASE cs_demo
USE Customer_view
BROWSE
```



Troubleshooting Slow-Opening Remote Views

Problem: When I open a remote view with a large number of records, FoxPro seems to fetch *all* the records causing an "ODBC driver is busy" error when I try to add a new record. How can I prevent this?

Solution: When you open a remote view, Visual FoxPro fetches information from the server. This information includes a scan of the entire view. If the view has a lot of records, this takes some time. To prevent this, set the MaxRecords property lower using the CURSORSETPROP() function.

For Example:

CURSORSETPROP() MSS_HELP

Presenting Your Data

Local and Remote Views are powerful tools that you can use to access large sets of data in a fast and efficient manner. However, presenting data to your users does not stop with views. The information generated by views is raw and unformatted data, so your next step is to format this report in a way that is easy for your users to interpret and understand. You can do this with the Visual FoxPro report design tools.

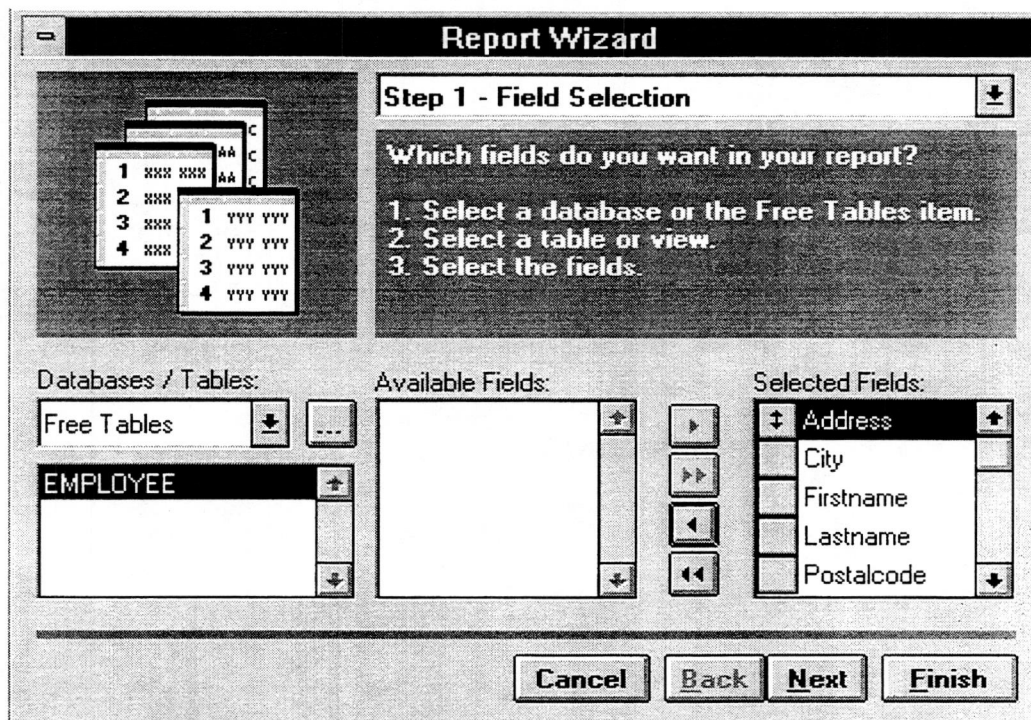
Visual FoxPro provides a set of tools that speed the development of complex reports. Included in this set of tools are Wizards that can be used to rapidly develop reports. Visual FoxPro provides Wizards that can generate single table reports, one to many reports, grouping reports, and graphs. These reports generate Visual FoxPro reports files (.FRX) that can be modified in the Report Designer.

The Visual FoxPro Report Wizards generate reports that can be maintained in the Report Designer. The Report Designer allows you to specify the many characteristics of a report including items to be printed, formatting information, grouping information, and summary information. The remainder of this section introduces you to the tools that can be used to generate formatted output from your Visual FoxPro applications.

Using the Report Wizard

After creating your views, you can create a report using the Report Wizard by selecting the report option in the Project Manager, and then selecting New. Selecting the new option presents you with the option of either using a Wizard or creating a report by hand. When you create your initial report it is wise to select the Wizard option, because the Report Wizard is a useful tool for creating a rough layout of your reports.

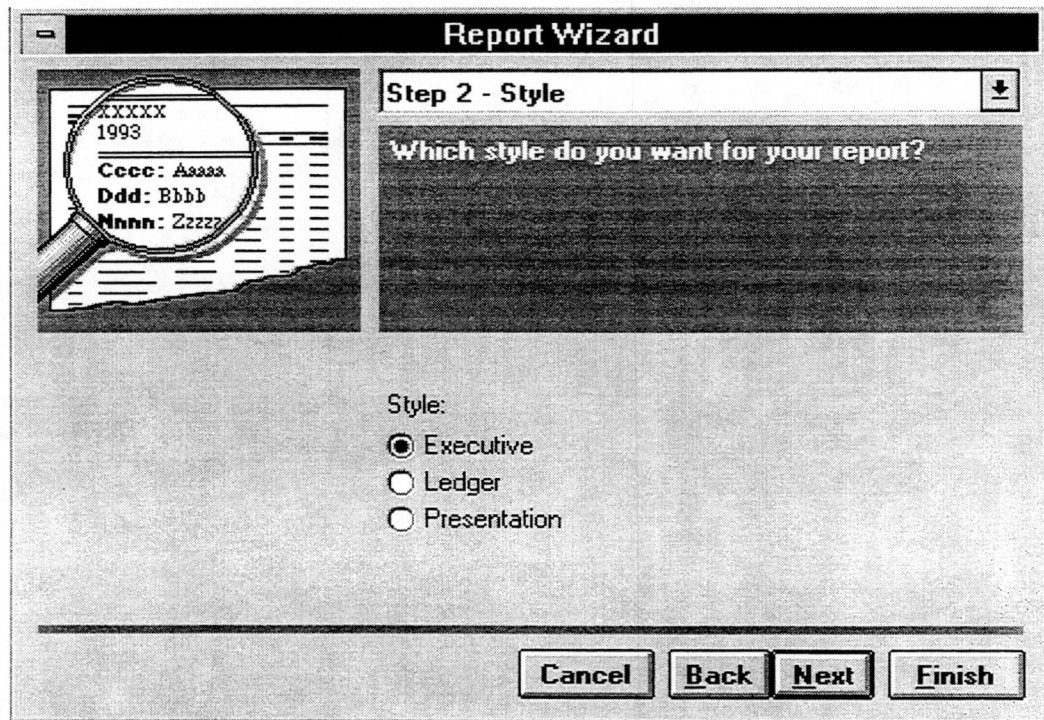
When you select the Report Wizard, FoxPro prompts you with the Table/Field selection dialog shown on the screen below. From this dialog you can select the tables or views to use in your report. Upon selecting a view or table for your report the Wizard lists the fields from that table or view in the available fields list.



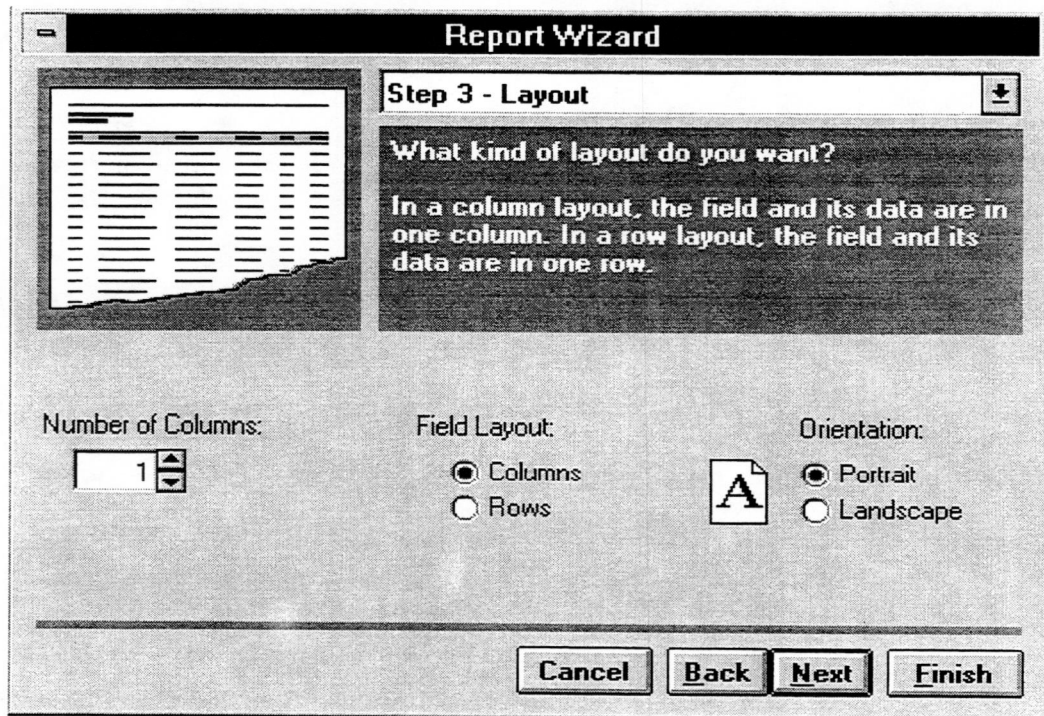
The next step to creating a report with the Report Wizard is to select the style of the report to be generated. The Report Wizard provides three standard report styles:

- Executive
- Ledger
- Presentation

The screen below shows the style selection screen of the Report Wizard:



After you select the style of your report, you can select a few more formatting options for your report. The next figure shows the Report Wizard Layout screen:



From this screen, you can alter the following attributes of your report:

- The number of columns to be created
- Whether to display the report data in columns or rows
- Whether to output the report in vertical or horizontal perspective

The final item you can specify for your report is the sort order. This is done with the *Sort Order* screen of the Report Wizard shown below:

Report Wizard

Step 4 - Sort Order

How do you want to sort your records?

Records will be sorted according to the order of the selected fields. You may select up to three fields.

Available Fields:

Address
City
Firstname
Lastname
Postalcode
State

Selected Fields:

☒ Ascending
☐ Descending

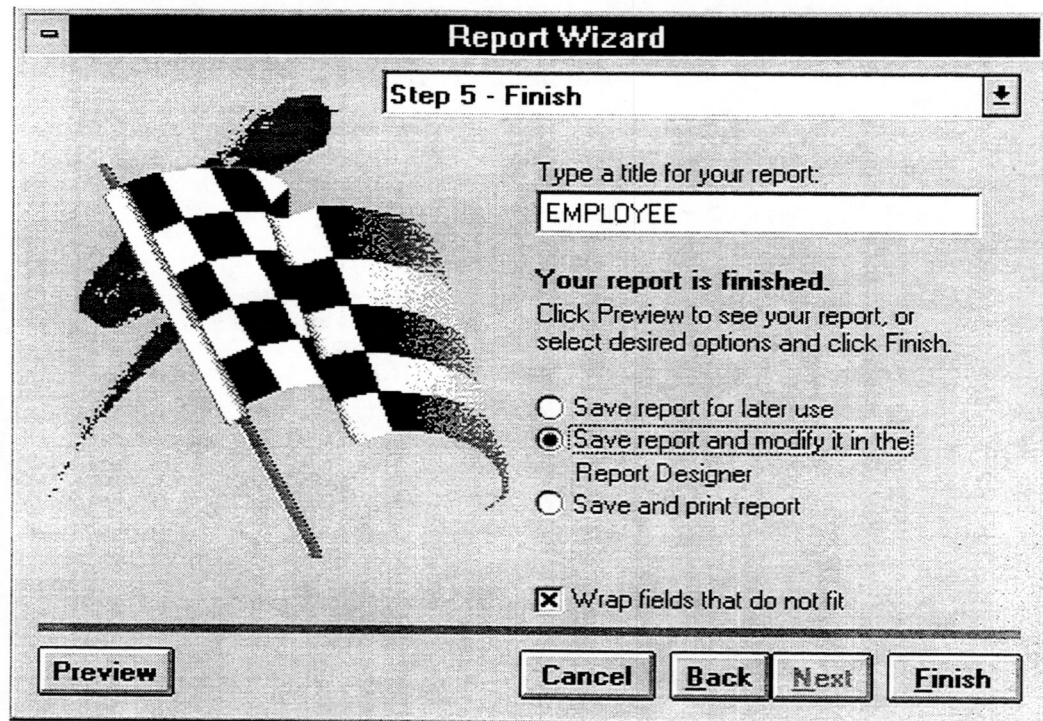
Buttons: Cancel, Back, Next, Finish

The Sort Order screen allows you to order your report according to the order of fields you select from your report.

WARNING! When you specify a sort order from the Report Wizard, FoxPro adds extra indexes to your files.

Upon specifying all of the various report options, you can now go to the finish screen and complete the process of creating your report. You can specify a few final options on the finish screen for the Report Wizard before finishing your report.

The last options that you can specify on the Finish Screen are a report title and whether and the wrap fields that do not fit option. If you do not select the wrap option, the Report Wizard drops the fields that do not fit from your finished report. The screen below represents the finishing screen for the FoxPro Report Wizard:



The image shows a 'Report Wizard' dialog box, specifically 'Step 5 - Finish'. On the left is a graphic of a checkered racing flag. The main area contains a text box for the report title, which is 'EMPLOYEE'. Below this, a message states 'Your report is finished.' and provides instructions to click 'Preview' or 'Finish'. There are three radio button options for saving the report: 'Save report for later use', 'Save report and modify it in the Report Designer' (which is selected), and 'Save and print report'. A checked checkbox labeled 'Wrap fields that do not fit' is also present. At the bottom are buttons for 'Preview', 'Cancel', 'Back', 'Next', and 'Finish'.

Report Wizard

Step 5 - Finish

Type a title for your report:
EMPLOYEE

Your report is finished.
Click Preview to see your report, or
select desired options and click Finish.

☐ Save report for later use
☒ Save report and modify it in the
Report Designer
☐ Save and print report

☒ Wrap fields that do not fit

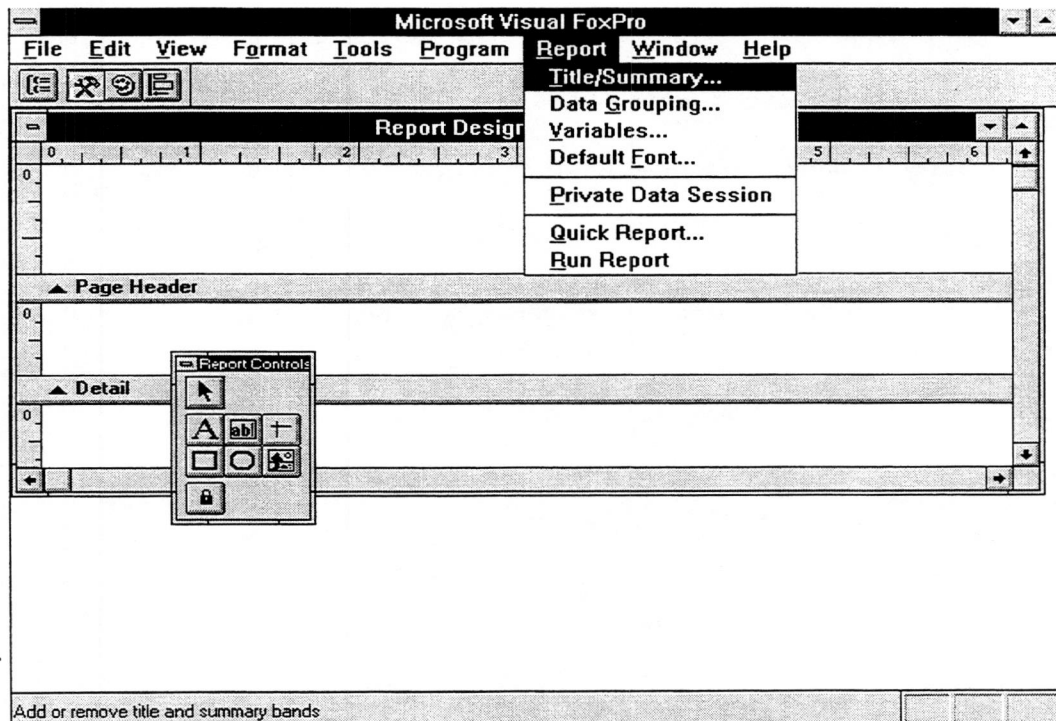
Preview Cancel Back Next Finish

Creating Reports with the Report Designer

As you learned earlier, you can use the Report Wizards to begin the initial layout of your reports. However, the Report Wizards do not always go the distance. This section explains in depth how to use the FoxPro Report Designer and its basic components.

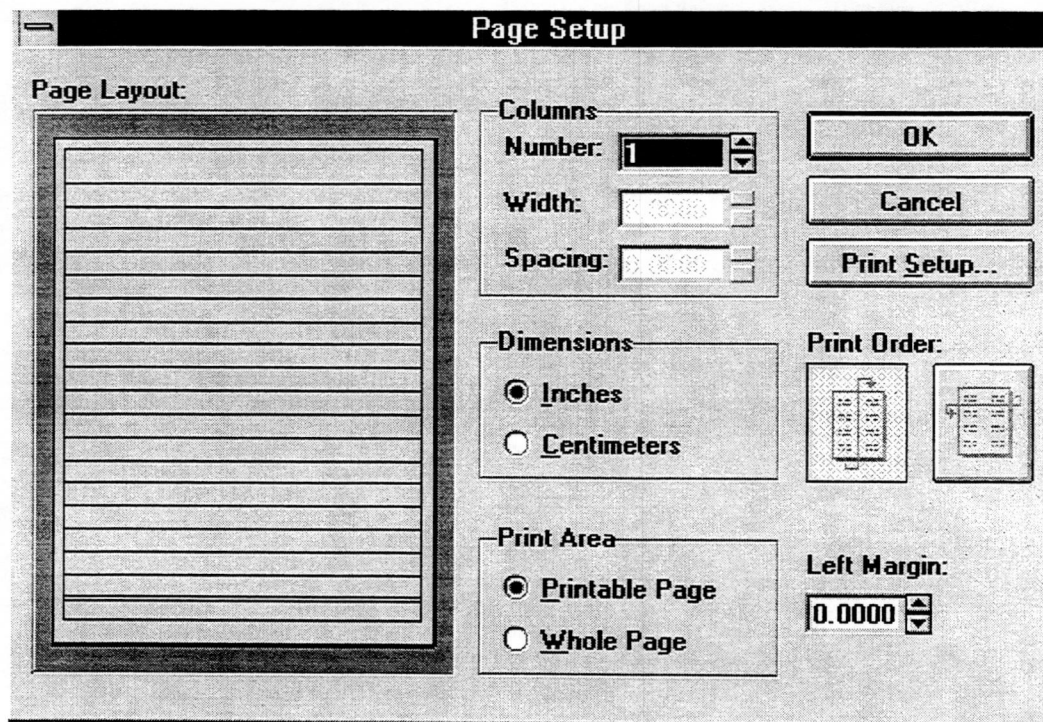
To activate the Report Designer, select the Reports option from the project manager outline and select New. This brings up the dialog that allows you to choose between using a Report Wizard or using the Report Designer to create a new report. To activate the Report Designer, select the Report Designer option.

When you select the report designer option, FoxPro presents the Report Designer design surface (see below). This is the tool you use to create your reports.



Specifying a Page Layout

The first item of concern is the layout of your report. You can control the layout of a FoxPro report by selecting Page Setup from the File menu. Selecting this option brings up the following dialog:



The Page Setup dialog allows you to control the following layout aspects of your report:

- Page size
- Page orientation
- Number and dimensions of columns
- The measurement dimensions of a page
- The default printer for the report

The Report Designer Toolbar

The next component of the Report Designer you need to become familiar with is the Report Designer toolbar. The report designer toolbar allows you to add and modify the objects that make up your report.

The options available from the Report Designer toolbar are:

- Pointer
- Label
- Field
- Line
- Box
- Rounded Box
- Picture

The *pointer* is a selection tool. Use it to select the location for placement of a report object onto a report. Also, use it to select an already placed object for modification.

Use the *label* tool to create a text object. This text may be as simple as a character string, or it may be built from a complex character expression using the expression builder.

Use the *field* tool for placing fields on the report. After you place a field on a report form, double-clicking on the field allows you to edit the field properties and expression.

The *line* tool creates lines on the report form. To place a line, select a starting point with the crosshair cursor and click once with the left mouse button. Then, select an ending point with the crosshair and click again with the left mouse button. Once you place a line, double-clicking on it brings up the Rectangle/Line properties dialog. You can fine-tune the appearance of the line object from this dialog.

All graphic objects such as lines, boxes, and pictures use the Windows standard sizing and moving conventions. Sizing and moving objects are discussed in a later section.

You may place a box on a report form using the *box* tool. To place a box, select a starting point for the upper left corner of the box with the crosshair cursor and click once with the left mouse button. Then, select an ending point for the lower right corner of the box with the crosshair and click again with the left mouse button. This places and sizes the box. Once you place a box, double-clicking on it brings up the Rectangle/Line properties dialog. You can fine-tune the look and behavior of the box object from this dialog.

The *rounded box* tool is like the box tool in all respects except that the corners of the box are rounded. You place and size it in the same way as a standard box. Once you place a rounded box, double-clicking on it brings up the rounded box properties option dialog. You can alter the appearance of the rounded box from this dialog.

The *picture* tool allows you to place a graphic on your report.

The Report Menu Options

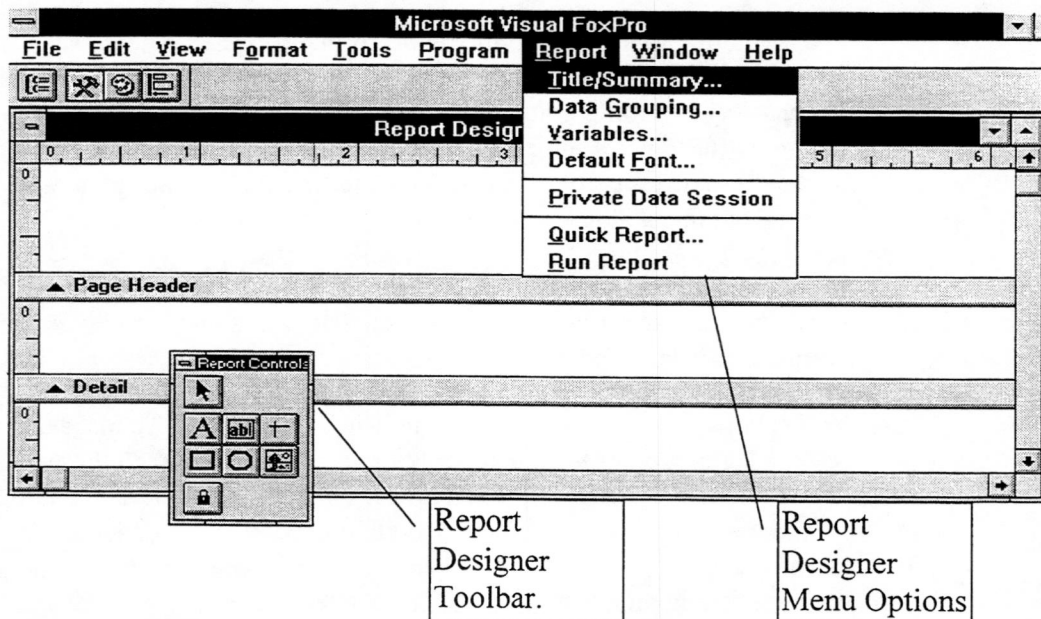
The report menu options allow you to control other options for your report. The options controlled by this menu are:

- Title/Summary Information
- Data Groupings
- Report Variables
- The Report's default Font

There are also a few options under on this screen that are activity-based report options. These options and their functions are:

- Private Data Session
- Quick Report - Allows you to rapidly create a basic report.
- Run Report - Allows you to preview your report.

The following figure points out the various components of the Report Designer:

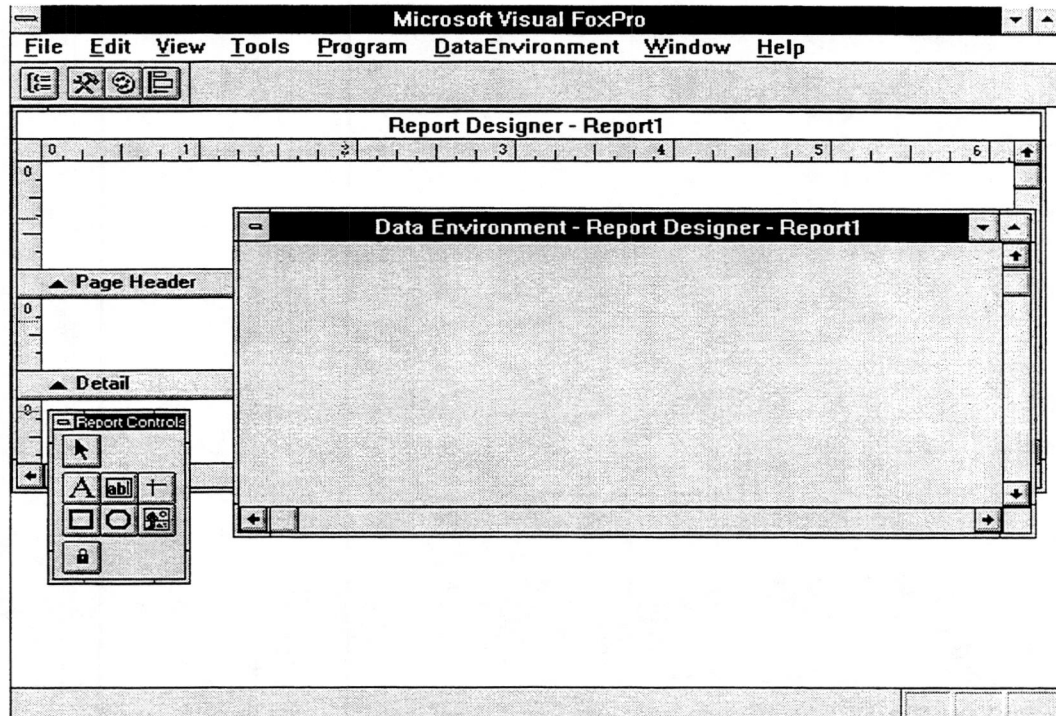


Report Data Environment

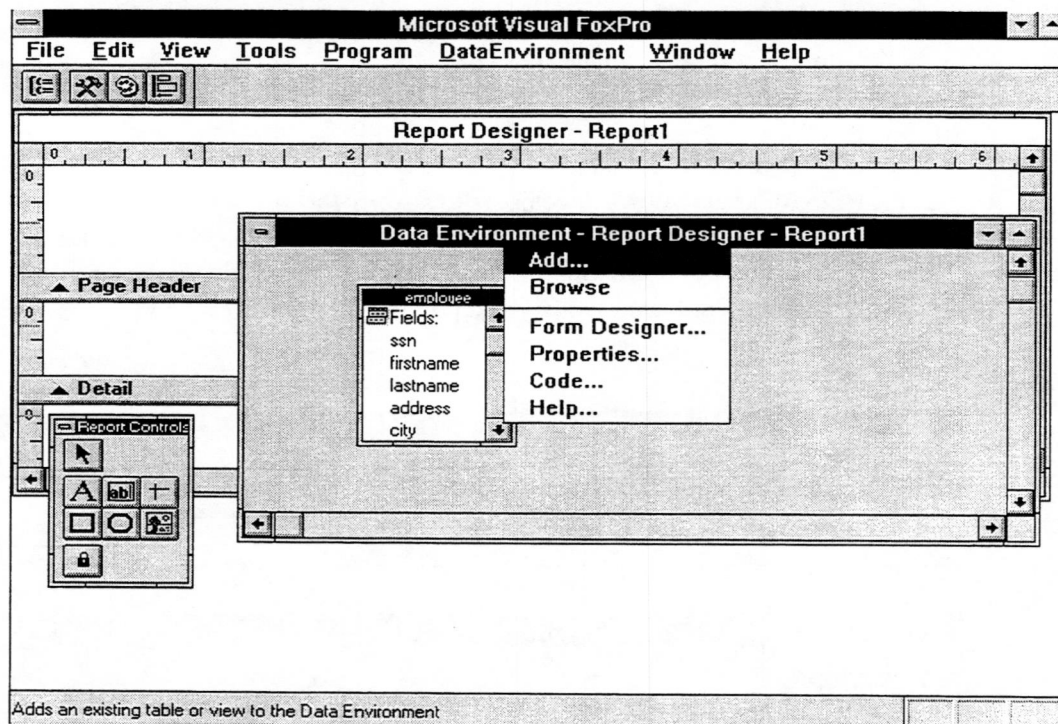
The last component of the Report Designer to explore is the report data environment. FoxPro 2.x provided an option that allowed you to save

something called the data environment when you created reports. However, saving this environment in FoxPro 2.x sometimes caused unforeseen problems.

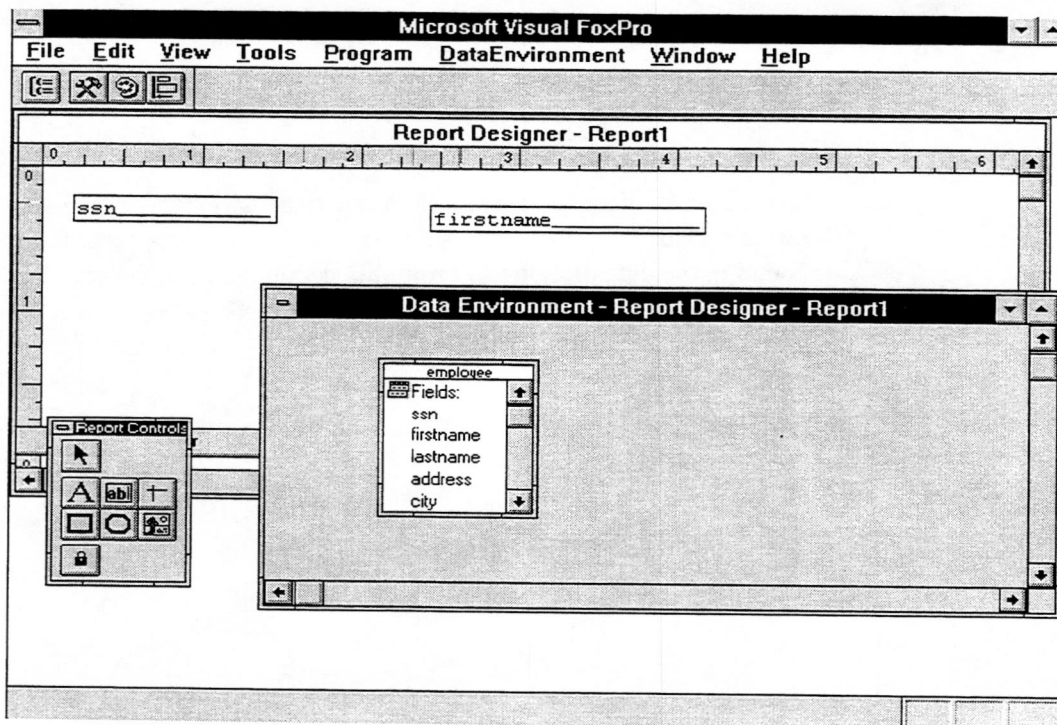
Visual FoxPro has a radically improved data environment tool. The Data Environment allows developers to specify the tables that are used for a report by placing them in the Data Environment. You can activate the Data Environment by clicking the right-mouse button and selecting Data Environment from the activated popup. This activates a dialog similar to the one below:



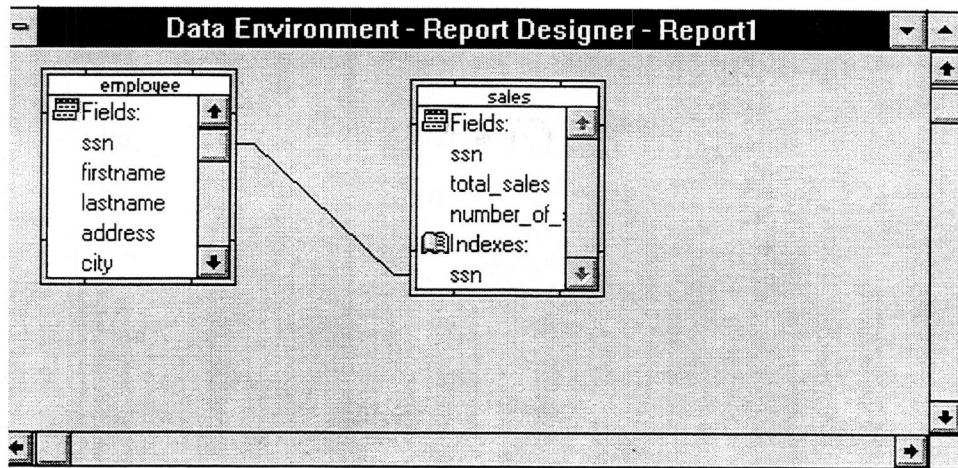
The Data Environment is very similar to the data dictionary tool. You can right click on the Data Environment to select options from a popup similar to the one found in the data dictionary. From this popup, you can choose to add tables to the Data Environment, add code to the report, or browse a table from the data environment. Right-clicking on the Data Environment activates the popup displayed below:



Selecting Add from this dialog activates another dialog that allows you to select from a list of tables to be added to the environment. The following example shows the Data Environment with a single table. Note that a useful feature of the Data Environment is its ability to drag fields from the data environment to the report design surfaces.



Another interesting aspect of the Data Environment is its ability to create relationships between tables. The following figure demonstrates a relationship between two tables in the report Data Environment surfaces.



Adding Objects to Your Reports

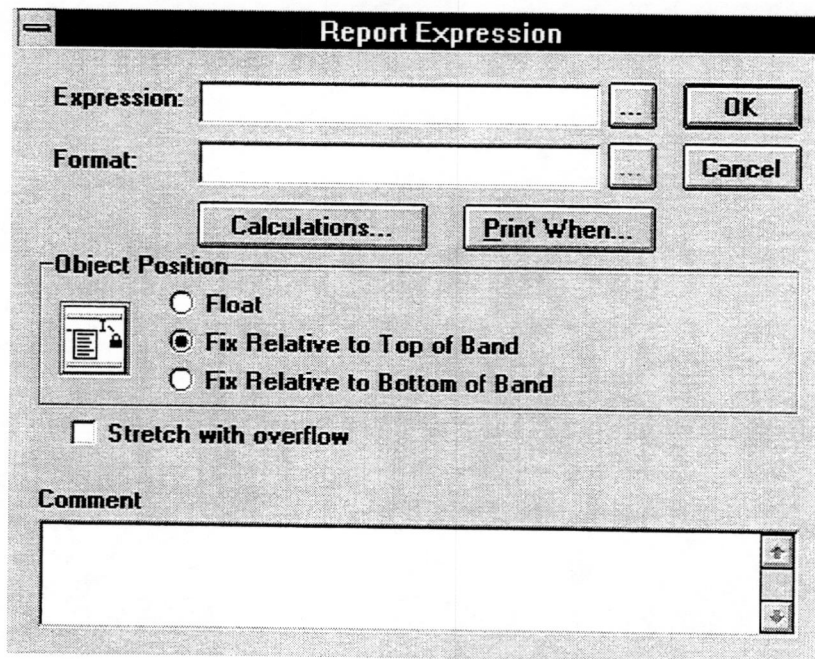
The Report Designer provides two methods of adding data objects to your reports. The first method is to drag objects from the data environment and drop them on the report design surface.

The second method is to select the field tool from the Report Designer toolbar. You already looked at how to drag fields from the Data Environment, now look at how to add objects with the field tool.

To add an object with the field tool, you need to do the following:

1. Select the field tool in the tool palette.
2. Highlight the area on the report surface where your object is to be sent.
3. Provide the dialog with the name of the field to be displayed or an expression to be printed.

After you complete the first the two steps, FoxPro opens the Report Expression dialog box:

The image shows the 'Report Expression' dialog box in FoxPro. It has a title bar with the text 'Report Expression'. Inside, there are two text input fields: 'Expression:' and 'Format:'. To the right of each field is a button with three dots (...). To the right of the 'Format:' field are 'OK' and 'Cancel' buttons. Below these are two buttons: 'Calculations...' and 'Print When...'. Underneath is a section titled 'Object Position' which contains three radio button options: 'Float', 'Fix Relative to Top of Band' (which is selected), and 'Fix Relative to Bottom of Band'. Below this section is a checkbox labeled 'Stretch with overflow'. At the bottom is a 'Comment' label followed by a large text area with vertical scroll bars.

This dialog allows you to specify the field to be placed on the report. You can define the name of the object to be placed on the form by typing in the name of the field to be printed prefixed by the name of the table in which this object resides. For example, *employee.ssn* represents the *ssn* field from the *employee* table.

You can also specify a field for a report by clicking on the ellipsis next to the Expression entry field. This brings up the FoxPro Expression builder, which is a more friendly interface for specifying fields.

In addition to adding fields to your reports, you can also use the toolbar to add visual components such as lines, boxes, graphics, and labels. To add any of these other components, simply select the desired control and highlight the region on the report that contains the object.

Other Report Components

From the Report menu option you can specify extra options for your reports. These options include:

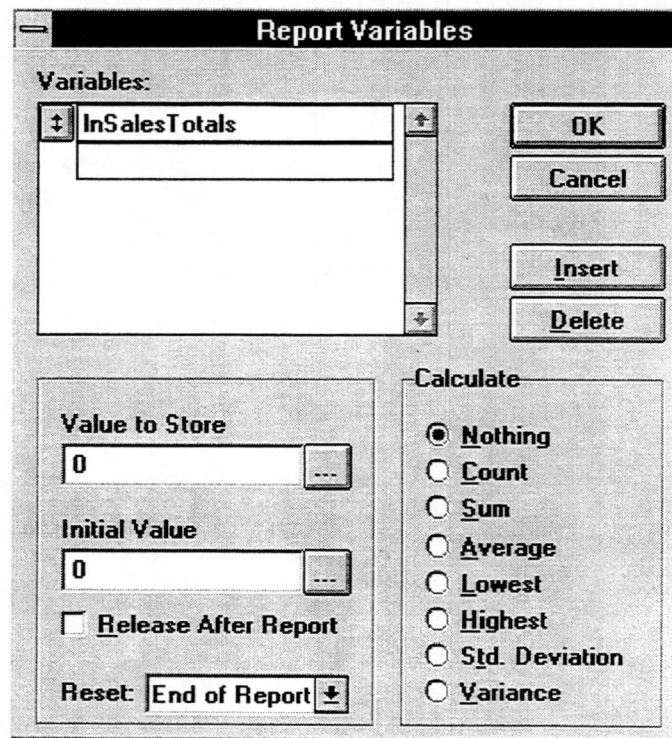
- Report Variables
- Data Groupings

Report Variables

Report variables are unique to the report writer. These special variables are created when you run the report.

A report variable is created when you run the report. It is released when you finish the report. This means that the values held in the variable are undefined outside of the report. A program, for example, that calls a report cannot reference a report type variable.

You can create a report variable by selecting the Variable option from the Report menu popup. This brings up the Report Variable dialog box shown below:



The **Report Variables** dialog box is used to define and manage report variables. It features a list of variables, with **InSalesTotals** currently selected. To the right of the list are buttons for **OK**, **Cancel**, **Insert**, and **Delete**. Below the list, there are two main sections: **Value to Store** and **Calculate**.

Value to Store section:

- Value to Store:** A text field containing **0** with an ellipsis button to its right.
- Initial Value:** A text field containing **0** with an ellipsis button to its right.
- Release After Report:** An unchecked checkbox.
- Reset:** A dropdown menu currently set to **End of Report** with a small arrow button.

Calculate section:

- Nothing** (selected with a radio button)
- Count** (radio button)
- Sum** (radio button)
- Average** (radio button)
- Lowest** (radio button)
- Highest** (radio button)
- Std. Deviation** (radio button)
- Variance** (radio button)

From this box you may add, change, or delete report variables.

The Delete option destroys the currently highlighted report variable. This option is grayed out if no report variables are currently defined.

The first thing you should do to create a report variable is name the report variable by placing a name in the Variables name box. This name must meet the requirements of a standard FoxPro variable name.

You may wish to store a value to this new report variable upon creating it. To do this, click on the ellipsis next to the Value to Store field. This brings up the standard FoxPro Expression Builder to help you assign a value of the appropriate type.

The Initial Value field differs from Value to Store in that while Value to Store specifies a value for the variable when it is *created*, Initial Value specifies a value for the variable each time it is *reset*.

You can reset a report variable's value at certain times during report execution. You can always reset a variable at either of the following two points:

- End of Page
- End of Report (default)

In addition, if you have used any grouping bands, the Report drop-down list also contains the names of those bands. Each grouping band becomes another point at which you may reset a report variable.

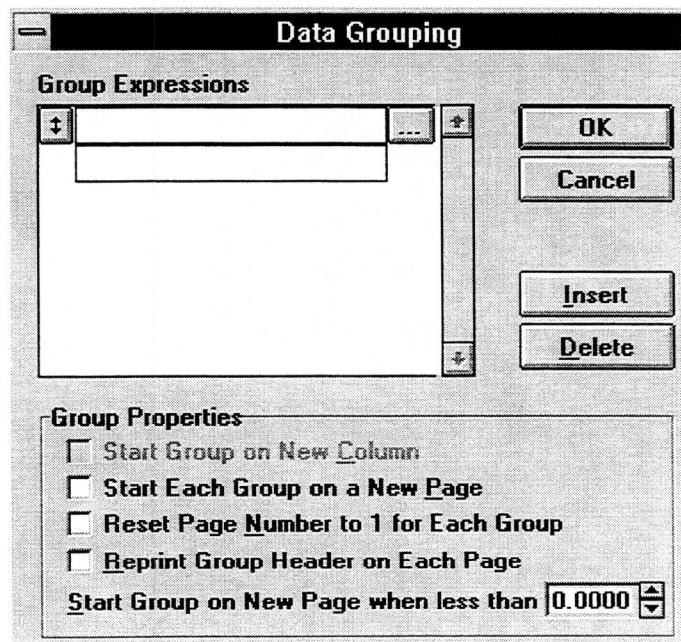
On the right side of the Variable Definition dialog, there is a set of radio buttons. Use these for numeric report variable calculations. They simplify your work when you create report variables since they allow you to do the most common calculations in a report. As you can imagine, it can take considerably more work to set up a variance calculation yourself (though you certainly can if you want to).

Click Change to modify the definition of the report variable that is currently highlighted. If no report variables currently exist, the Change button is disabled.

Data Groupings

The group band setup affects how the detail band information is presented. As the name implies, the Group band forces the detail bands to be printed logically broken up into groups. For example, if you want a list of customers by state, sort by state and customer name, and group by state. The result is a heading and subtotal of sales by state, yet within each state the customers are listed in order by customer name.

To create a new group, select the Data Grouping option from the Report menu. This action brings up the Data Grouping dialog as shown below. If you define any data grouping, it is listed in the Group list box on the top left side of the dialog box.



If you want the data grouping to cause special page or column breaks, select the appropriate box in the Group Properties section of the Group Info dialog box. By default, no unusual breaks occur. Rather, the information contained in the data group header and footer is printed. You can repeat the group header at the top of each new page that is spanned by a group by selecting the Reprint Header on Subsequent Pages check box.

You can prevent *orphaned* group headers by using the final option on the Data Grouping dialog box. This option forces a new page if the group break occurs too close to the end of page. The tolerance for this option is measured from the header rather than from the footer. If set to zero, orphan checking is disabled.

Clicking Group takes you to the FoxPro Expression Builder, where you can construct a grouping expression using point and click. After you construct an expression, it is a good idea to click Verify to check the validity of your expressions.

Remember that for the data grouping expression to be meaningful, the table must be ordered by that same expression. At runtime, a new group begins each time the group expression changes.

Be careful to not use the terms *group* and *sort* interchangeably. Users frequently request a report *sorted* in some way when in fact they mean they want it *grouped*. This usage can be a source of much confusion. For example, when users request that a report be sorted by state (or any other entity that occurs more than once in the report data), they usually mean they want it grouped by state and then sorted by some other meaningful attribute *within* the state grouping.

Introduction to Object- Oriented Programming

Objectives

One of the biggest changes found in the Visual FoxPro environment is the ability to create re-usable components using object-oriented technology. This section discusses the new terminology found in the object-oriented development world and how you can begin implementing it in your Visual FoxPro applications.

In this section you:

- Learn object-oriented terminology.
- Learn why object-oriented programming is important.
- Save form components using the SAVE AS CLASS... option.
- Add custom methods and properties to your objects.
- Use your custom objects in your forms.

Understanding Object-Oriented Programming (OOP) Terminology

The first step to understanding any new technology is to understand the terminology of that new technology. Object-oriented programming is full of new terminology. Defined below are many of the key terms found in object-oriented programming.

Classes	Blueprints that define the properties and methods of your objects.
Properties	Attributes of the objects. Object variables.
Protected Property	A property of an object that can only be seen and modified by the object itself.
Methods	Procedures or functions that implement the object's functionality.
Protected Method	A method that can be called only within the object itself.
Class Library	Collections of classes.
Instantiation/Instance	Runtime creation of an object in memory derived from a class definition.
Encapsulation	The act of including all data and code required by an object within that object. This makes that object self-contained.
Inheritance	The process whereby each class obtains default data (properties) and behavior (methods) from its parent class. Inheritance is dynamic; it is not limited to what is true at the time you create a child class. If you make changes to a parent class, its present <i>and</i> future children automatically inherit those changes. This causes bug fixes, for example, to "ripple" through a class hierarchy.
Subclassing	To create a new class from an existing class, the new class inherits all the properties and methods of its parent class. You can add new properties or methods to the new class, or change (override) properties or methods inherited from the parent.

Class Hierarchy	The parent-child relationship between classes.
Polymorphism	The ability to associate different properties and methods that have the same names with various objects. For example, visual objects in an application could each have a “draw” method which might be implemented differently for each object, yet performs the same task of redrawing the object. Thus, you could command any of these objects to “draw” themselves without caring that the functionality is implemented in different routines.

So What is the Point?

At first glance, OOP may seem like smoke and mirrors—little more than new, trendy terminology to obfuscate the structured programming techniques you are probably already familiar with. After all, a method is just a function, and a property is a kind of variable, right? True, but there are key differences in the conceptual framework, the packaging, and the powerful implications of inheritance.

In previous versions of FoxPro, as in other non-OOP languages, it is possible to achieve a respectable level of data hiding, modularity, and code re-usability through strict adherence to certain programming standards, techniques and tricks. But OOP provides built-in support for these objectives, making them much simpler to achieve. As a bonus, it *greatly* enhances code re-usability through the mechanism of inheritance.

With OOP, you do not need to write the same code more than once, nor are you exposed to the maintenance hazards associated with cutting and pasting codes between applications and routine libraries, that, when you think of it, is code *copying*, not true code *reuse*.

The entire purpose of object-oriented programming is to create reusable components for your applications. Object-oriented programming is not new. If you have been developing FoxPro 2.x applications, you are probably familiar with the process of designing reusable components using function and procedure libraries. Object-oriented programming takes this one step further. You can now create reusable components that allow you to change a parent object, and all of your changes are rippled to the children created from that parent.

Establishing a Firm Foundation

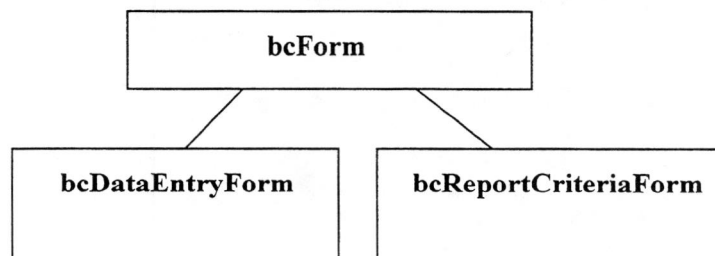
The first step to incorporating object oriented programming into your development methodology is to establish a set of base components that all of your applications will be developed from. This is commonly known as a set of foundation classes.

Foundation classes are the “widgets” that you assemble in your applications. In essence they are the nuts, bolts, pulleys, and gears that make up your applications. In order to create a set of foundation classes you need to spend some time up front determining the types of components to use in your applications. Commonly, you begin this by looking for common features found in all applications. For example, Windows applications have in common classes that you can use to open and save files, print files, and show messages on the screen. In the Visual FoxPro environment; your foundation classes would probably consist of generic data entry classes (Add, Edit, Save, Cancel), pre-formatted text entry fields (SSN, Phone #, Zip Code, Addresses, Etc.), common dialogs for printing reports and labels, and many others. Upon determining some of these classes that you wish to place in your libraries you need to sit down and begin creating your class libraries.

Foundation First Footsteps

The first step to beginning your applications is to create the most basic of classes for your component development. These are commonly known as primitives. The primitives in a Visual FoxPro class library commonly begin with the Visual FoxPro base classes. So, the first step is to subclass the Visual FoxPro base classes into classes in your libraries. A common naming convention for these base classes is to prefix them with the letters bc and then name the class with the Visual FoxPro base class name. So the Visual FoxPro form base class will be named bcForm.

Upon subclassing all of Visual FoxPro’s classes you can begin creating the classes that will become your Foundation Classes. Below is a diagram of some potential classes that can be created from the bcForm primitive.



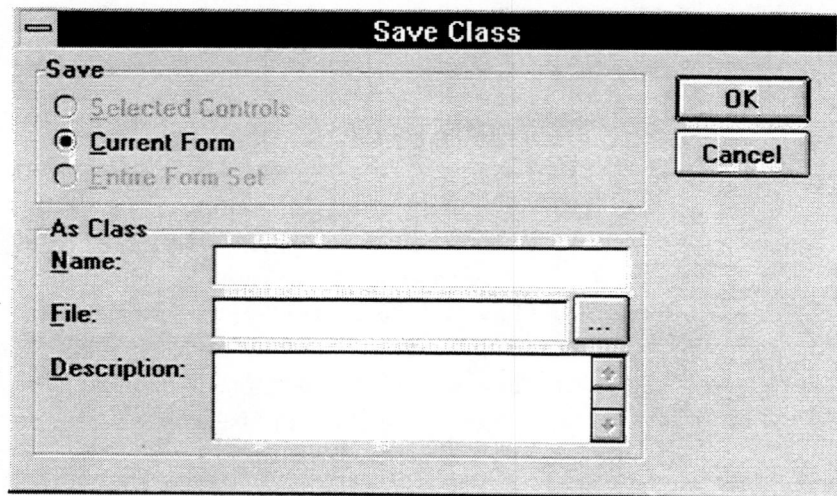
In the structure above, if you needed to make a global change to all forms, such as a background color change, you would go to the bcForm class and the change would be rippled through the bcDataEntryForm and bcReportCriteriaForm classes. This is where the power and need for a set of

foundation classes comes in. Now that you understand how to begin the inclusion of object-oriented foundations in your development, you can begin looking at the tools used to create these foundation classes. The remainder of this section is dedicated to the use of Visual FoxPro's object-oriented development tools.

Saving Form Controls as Classes

You can begin your first venture into the world of object-oriented programming from the Form Designer. The Form Designer has an option on the File menu that allows you to save forms and form components into a class library.

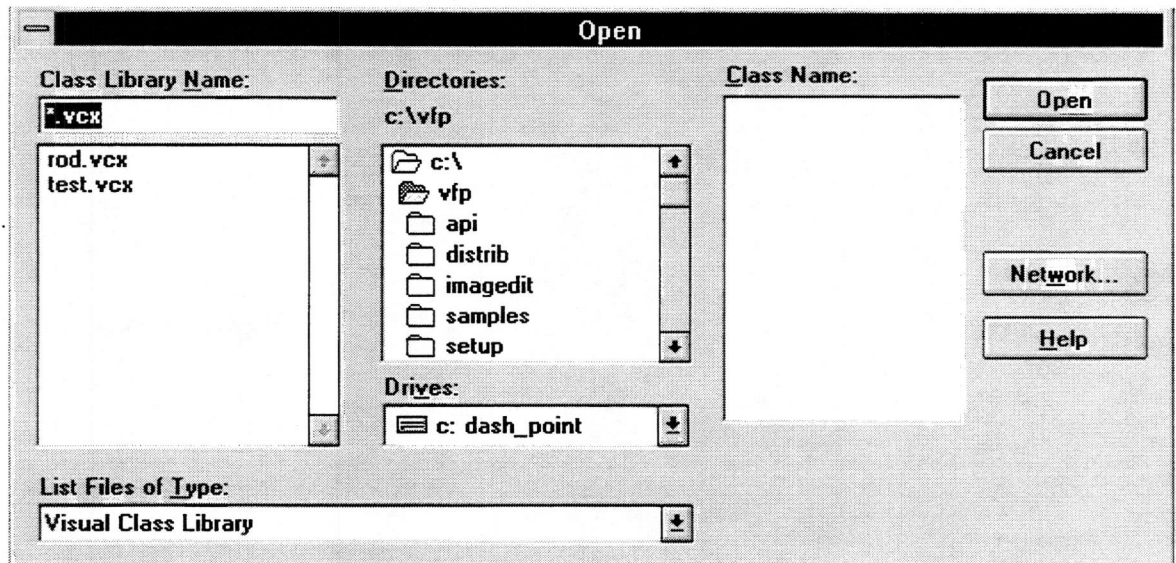
To save your form or form components as a class, select the object that you wish to save with the selection marquee, and select Save As Class from the file menu. After you select this option, you are prompted with a dialog requesting information about the class you are about to save.



This dialog prompts you for:

- The name of the new class.
- Whether to save the form, form set, or selected controls as a class.
- The name of the class library to save the class into.
- Some comments that are stored with the class.

Upon saving your class in a class library you can modify the class using a tool known as the Visual Class Designer. To modify your class, type: MODIFY CLASS in the command window. Upon typing this command, Visual FoxPro allows you to select a class to modify from a list of available class libraries.



Using the Visual Class Designer

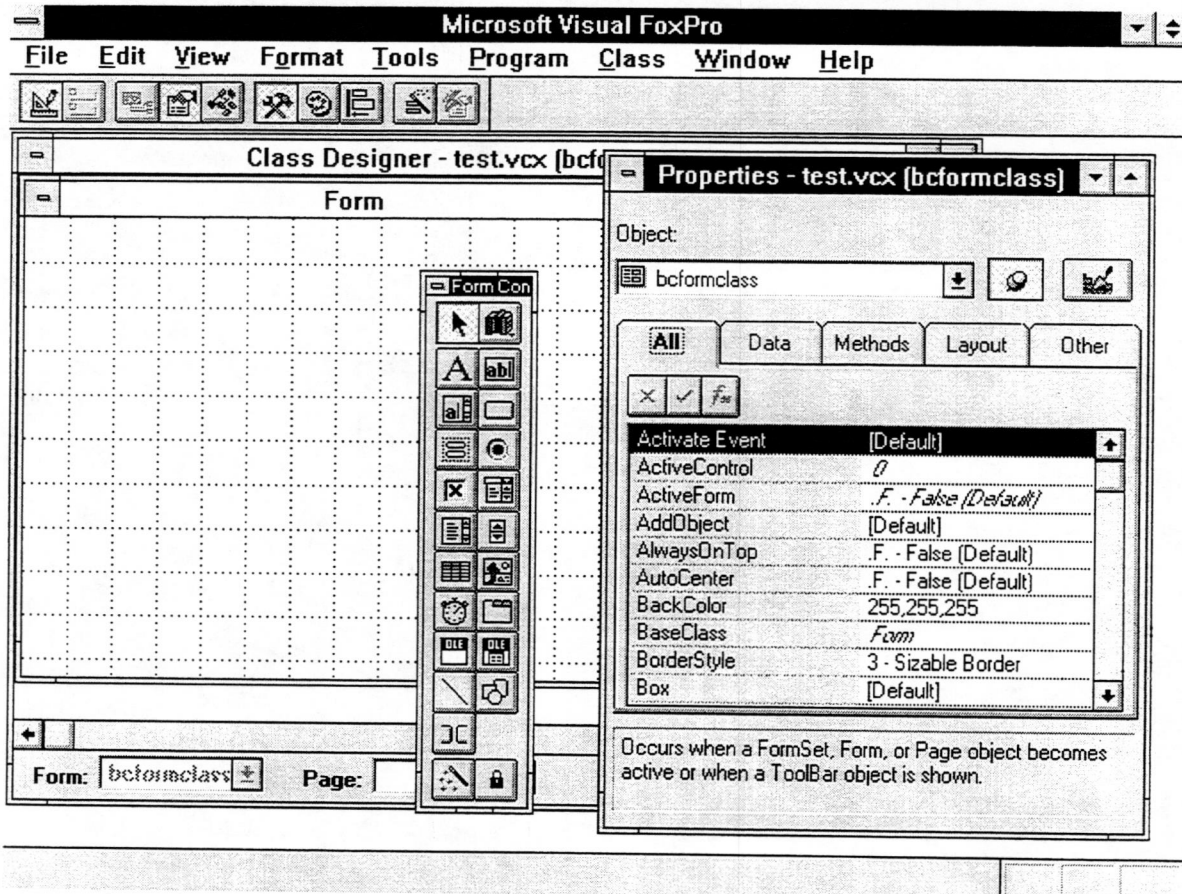
Upon selecting a class for modification, you are met with the Visual Class Designer. This tool allows you to specify various options for your class, and modify the following aspects of a class:

- Properties
- Methods
- Controls contained in the class
- An include file for the class
- Class definition properties

Adding Controls to a Class

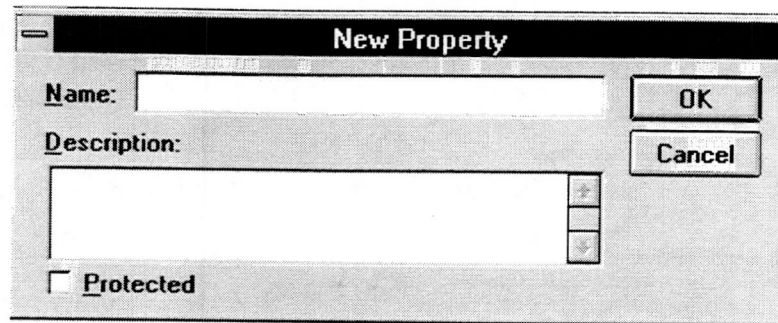
The Class designer is simply an extension of the Form Designer. The difference is that you can create reusable components in the class designer that become the basis of your application development. To add controls to your class definition, select them from the tool palate and add them to your form.

This procedure functions just like the Form Designer tool. The following illustration shows the Class Designer; notice the similarities with the Form Designer:



Adding/Modifying Class Properties

One of the features of the Visual Class Designer is its ability to add new properties to a class definition. Properties are the data components of an object; think of these properties as variables that are attached to an object. To add a new property to your class, select New Property from the Class menu. This presents you with the following dialog:



This dialog allows you to specify information about a particular property. The attributes that you can modify are:

- The name of the property
- A description of the property
- Whether or not the property is to be protected

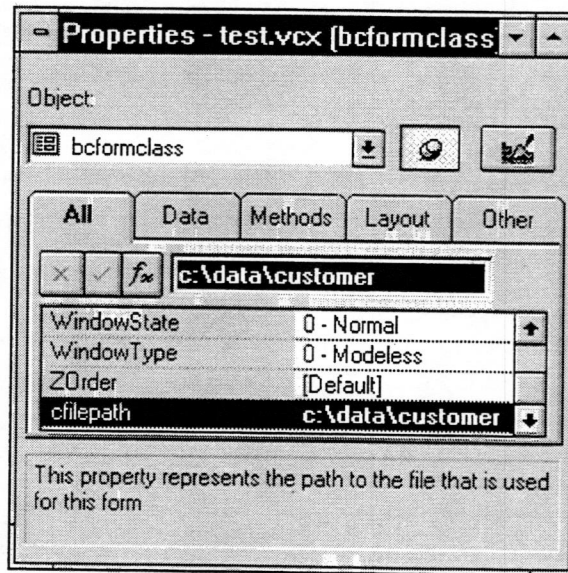
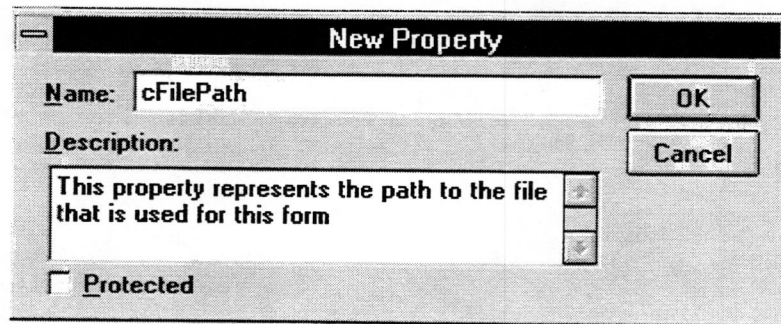
When naming your properties, make the first letter the same as the first letter of the data type of the property. For a review of the data type naming conventions, see Section 1 of this manual.

The description of the property allows you to specify text that is shown in the description section of the properties sheet. It is a good idea to describe each new property you are going to add. This allows you to further document your class definitions.

The last component that can be changed is a flag that determines whether this property is to be protected. Protected properties allow you to hide information inside your objects and protect that data from being modified from outside the object itself.

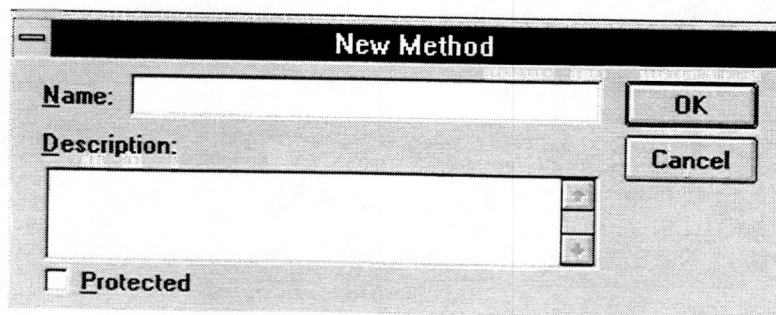
One of the design decisions you have to make in designing a class is what properties to *expose*. Properties that are not protected are exposed to the world outside the object. Exposed properties constitute part of the documented interface for your object that other objects use to interact with it. In general, expose properties that other objects find useful to query and change. Protect properties when you want to save a value for internal use only, or when you want to maintain tight control over how or when the value gets queried or changed, and what happens in response to that change. Protected properties can still be queried or changed by the “outside world” if you provide and document methods that can be called for this purpose.

The following illustrations represent a property definition and that property displayed in the properties sheet; notice the description text:



Adding/Modifying Class Methods

Another feature of the Visual Class Designer is its ability to add new methods to a class definition. Methods are the code components of an object. Think of these methods as library functions that are attached to an object. To add a new method to your class, select New Method from the Class menu. This presents you with the following dialog:

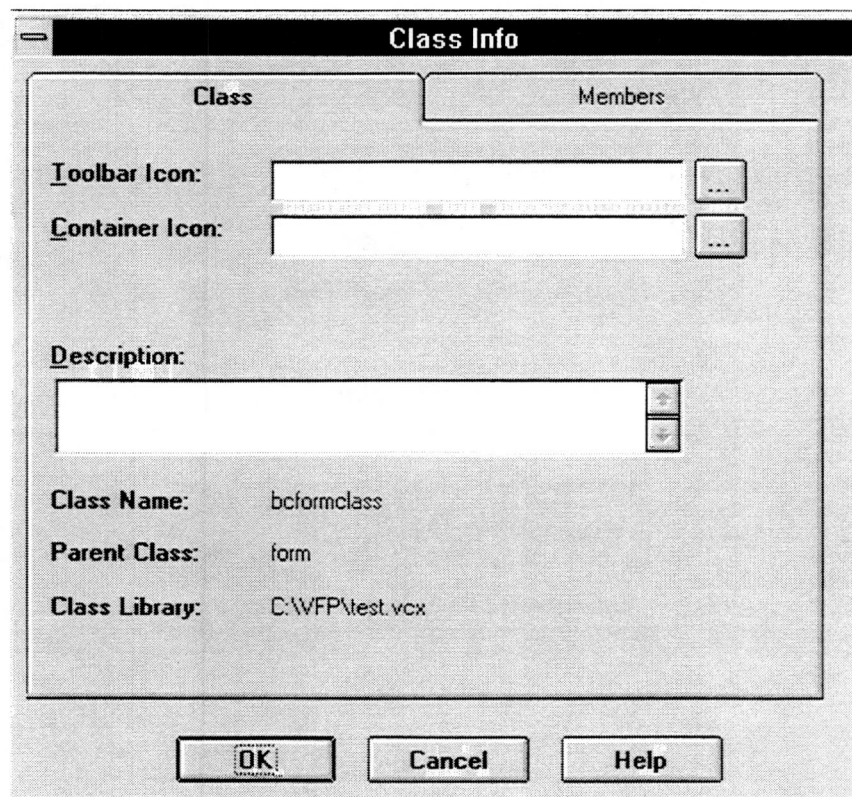


This dialog functions exactly like the new property dialog described above. After adding a new method, place code in this method by double-clicking on the method from the property sheet, or by double-clicking on the object in the design surfaces.

Setting Class Attributes

After adding all your controls, properties, and methods, you can now set a few final options for your class.

Selecting Class Info from the Class menu bar presents you with the following dialog:



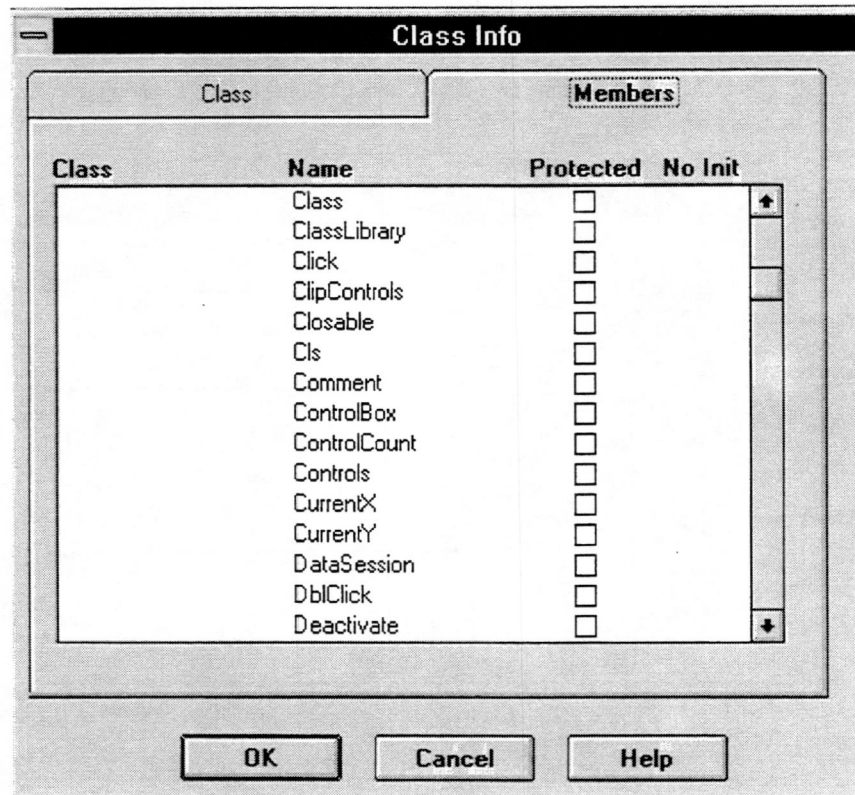
This dialog allows you to specify descriptive information about your class. You can specify:

- A toolbar icon for your class
- A container icon for your class
- A description of your class
- Whether a property or method is protected

The toolbar icon option allows you to specify an icon file that is used whenever your class appears in a form designer toolbar. The container icon allows you to specify an icon to be displayed in the Class Browser program.

You can also specify a description of the class. This is useful for documentation purposes.

The Class options dialog also allows you to protect any of the default properties for a class by using the members tab (shown below):



Putting it Together

Objectives

So far you have learned how to build component pieces for your applications, create forms, keep your data valid, create object-oriented components, and access client/server data. Here is how to put it all together.

In this section you:

- Use the Visual FoxPro Menu Designer.
- Create new menus.
- Specify menu options.
- Create a main program for your application.
- Compile your application into an EXE.
- Distribute your application with the Setup Wizard.

The Menu Designer

The first step towards finalizing your application is to create a menu. Visual FoxPro provides a tool called the Menu Designer that generates Windows menus. But, as with all development processes, you must first spend some time planning your menus.

Planning a Menu

The FoxPro Menu Designer is an excellent power tool for generating menu programs. It provides a rapid method for designing pull-down menus for Visual FoxPro applications. This tool relieves the developer of the time-consuming manual coding that is usually associated with building an application menu.

The Menu Designer (MD) is very much like the Form Designer. The MD uses a program generator (GENMENU.PRG) to generate native FoxPro menu source code.

The benefits of using the MD include reduced development time, ease of modifying menus, and the ability to quickly prototype an entire menu system. You can also alter the GENMENU program to meet your own requirements.

NOTE: Be sure to back up the original GENMENU.PRG if you decide to make enhancements to it.

The MD only constructs pull-down menus, the only types of menus that are standard on all FoxPro platforms. Pull-down menus are also the only types of menus that are event-based, and thus lend themselves to event-driven programming. A menu created with the MD is stored in a standard FoxPro table with an associated memo file. The table has an MNX extension, while the memo file has an MNT extension.

The program statements generated by the GENMENU.PRG program from the menu layout go to a file with an MPR extension that has the same base name as the menu layout. It is important to realize that this generated menu program only *defines* and *activates* a menu system or modifies the definition of an existing menu system. It executes little or no procedural code.

Your placement of the routines that can be called when the user invokes a menu option is very important. Remember that a DO command that is a response to a menu option is *not* called when the MPR is run. By the time the requested DO command is actually executed, the MPR program is no longer

running. Therefore, the routine called by the DO command is located and executed based on the environment at the time.

Visual FoxPro cannot find procedures and functions that are stored in the Cleanup code of the MPR program. This is because the MPR program is no longer running, and therefore is not in the calling stack. The key here is to put all routines that execute in response to menu hits in a place where they are found when those menu hits occur. This might be in a procedure file, in the startup PRG file, or as standalone PRG files.

Another option for procedures called with DO only, is to use the optional IN clause of the DO command, as in "DO MyPrg IN MyMenu.MPR". This permits you to package the routines with the menu. This approach, however, is clumsy and does not usually represent the best approach from a code organization standpoint. Besides, DO ... IN is generally the slowest way in FoxPro to execute a subroutine.

The MD builds menus that function exactly like Visual FoxPro's system menu. The top-level options are displayed across the top of the screen on a menu bar. Each option on this menu bar is called a *menu pad*.

When you select the menu pad with the keyboard or with a mouse click, a *menu popup* usually appears beneath the menu pad. This type of menu is known as a *pull-down* or *drop-down* menu because of its visual effect when selected. Incidentally, Visual FoxPro refers to these pull-down menus as *submenus*. Therefore, in Visual FoxPro, a menu popup and a submenu are one and the same.

The menu popup has options that are called *bars*. A menu bar, when selected, can execute a FoxPro command, a program, or another popup (submenu). Each level of submenus is staggered to one side or the other of the previous submenu or menu popup. This way, all options up the menu lines are visible. If selecting an option results in a submenu or dialog box, you should end the menu prompt in an ellipsis (...). This follows the Windows standards for menus, and is also the way FoxPro does it. (Look at the first three options under FoxPro's File system menu pad.)

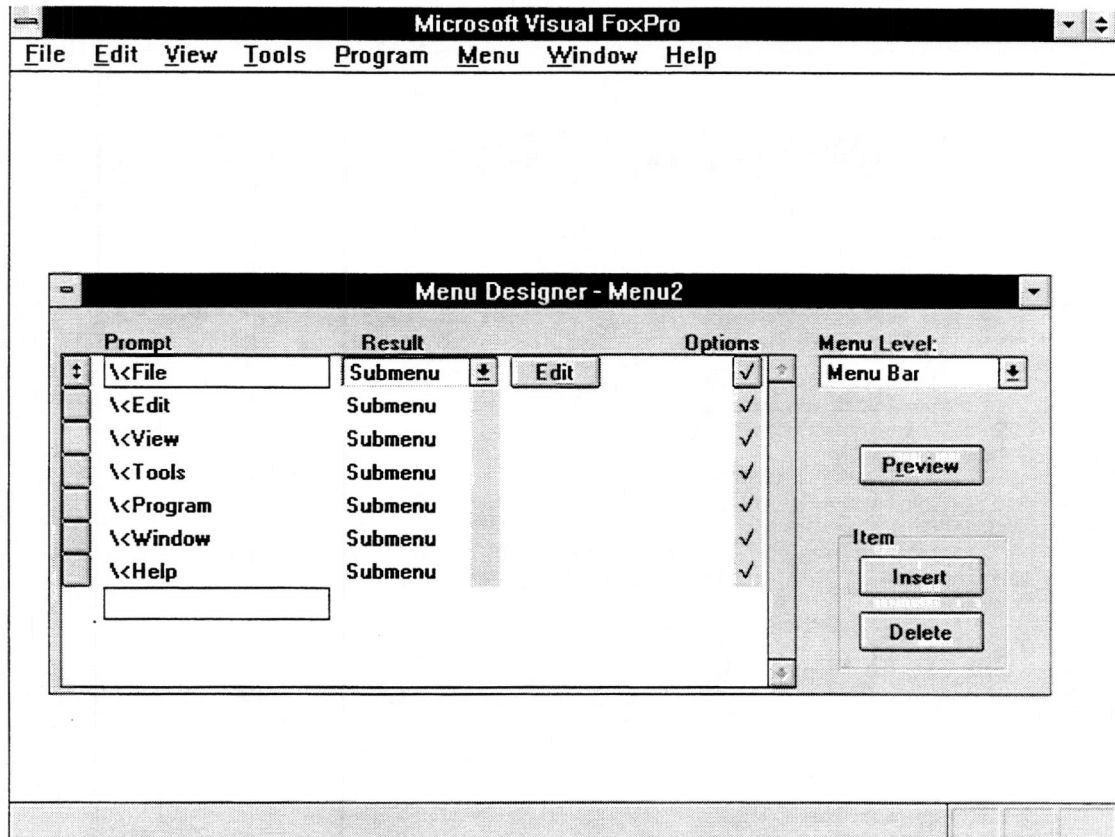
Creating a Menu Using Quick Menu

To create a Quick Menu from the PM, select New from the project window. Then select the Menu type from the Add File dialog box and select New. Be sure to specify the name of the new menu in the Save As Dialog Box and select ENTER. The Menu Designer loads with all fields empty.

Once the MD is started, FoxPro adds a Menu pad to the system menu bar. From Menu, select Quick Menu. FoxPro fills in the Menu Designer with the

default FoxPro menu options. This method provides you with an example of how the Menu Designer builds a menu program.

Try your menu while you are in the Menu Designer by clicking the Preview option in the menu dialog screen. This option is very helpful in determining how the menu looks at runtime.

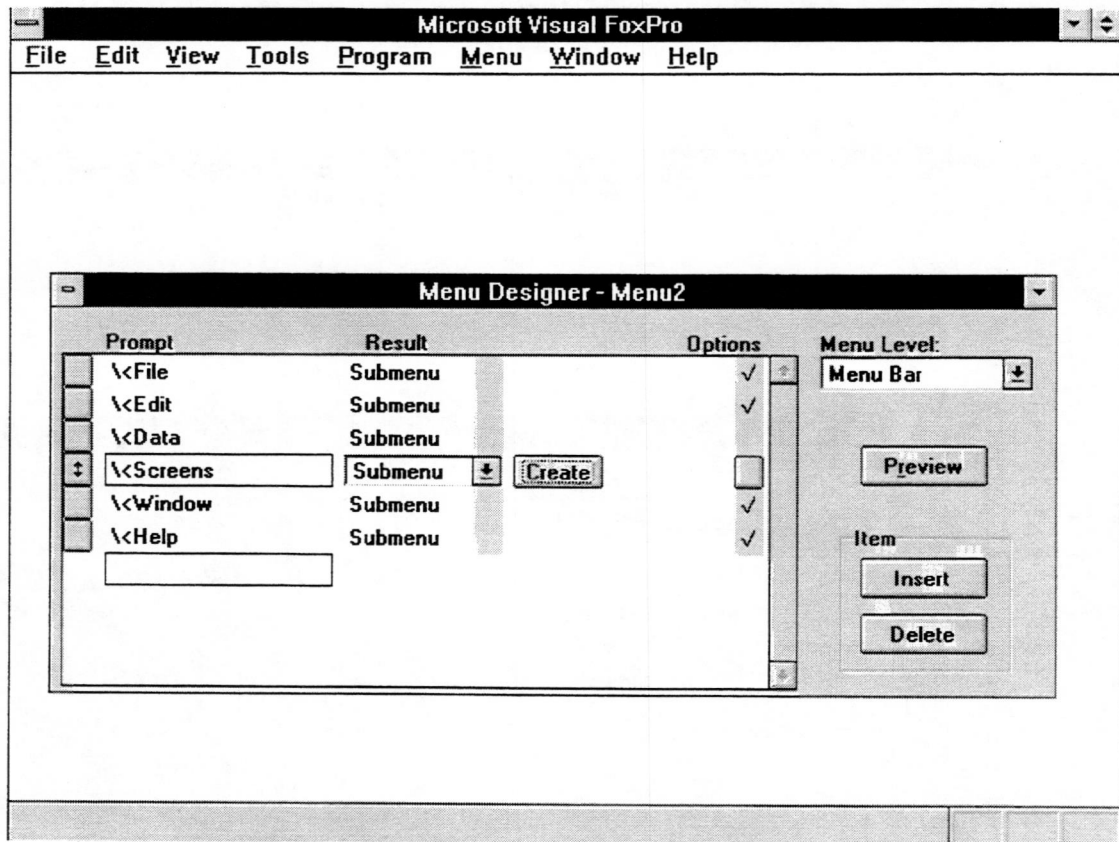


Once you have these defaults defined in the Menu Designer, you can delete, modify, and add other options to the menu. It is the developer's responsibility to design the code that is executed when a menu selection is made. The next section covers this.

Adding Application-Specific Options

Each menu pad is represented by a horizontal line of fields in the MD. Specify a prompt, a result, and several options for each menu pad. Add menu hot keys for each menu pad by placing a "\<" in front of the desired hot key letter. To initially disable a menu pad, place a "\<" at the beginning of any prompt name.

Each menu pad must have a result that executes when the menu pad is selected. The result choice can be a command, a submenu, a bar number, or a procedure.

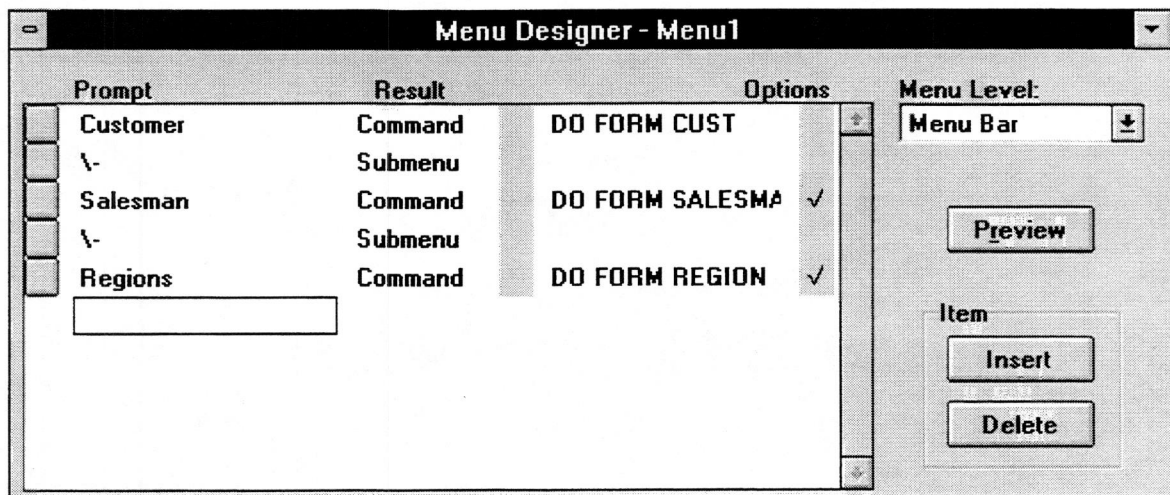


The *command* result option causes a text box to appear to the right of the result field.

Enter a valid FoxPro command into this text box. This includes a DO command that executes a program in the current path. This command is executed when you select that menu item.

NOTE: FoxPro does *not* do any error checking when you enter a command in the text box. The developer is responsible for the validity of the entered command.

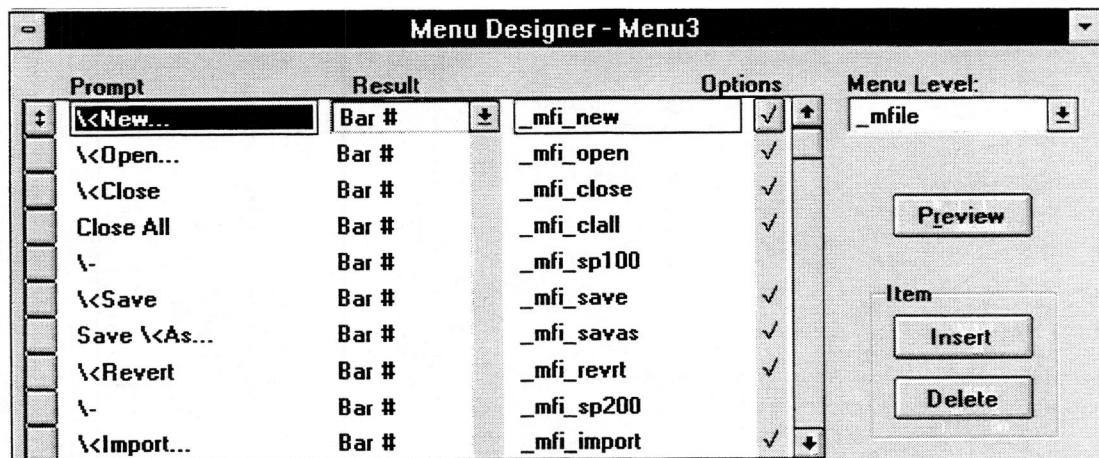
The *submenu* result is the default. The push-button to the right of the result field can read Create or Edit, depending on whether the submenu already exists or not. When you click this button, a submenu design window appears. This menu looks exactly like the main MD design window. The main difference between the two design windows is that the Submenu Design window builds popups (submenus), and the Menu Design window produces a menu bar.



Additional prompt options may be needed when designing submenus. For example, you may need to divide or group submenu options on your pull-down menus. Place “\” in the prompt field by itself to do this. This inserts a divider line for that menu option in your pull-down menu.

The *bar #* result causes a text box to appear at the right of the result field. Enter the bar name of a predefined FoxPro menu option in this text box, such as the editing options shown in the following figure.

The FoxPro Language Reference manual contains a complete list of all menu pad and bar names. The SYS(2013) command also provides such a list.

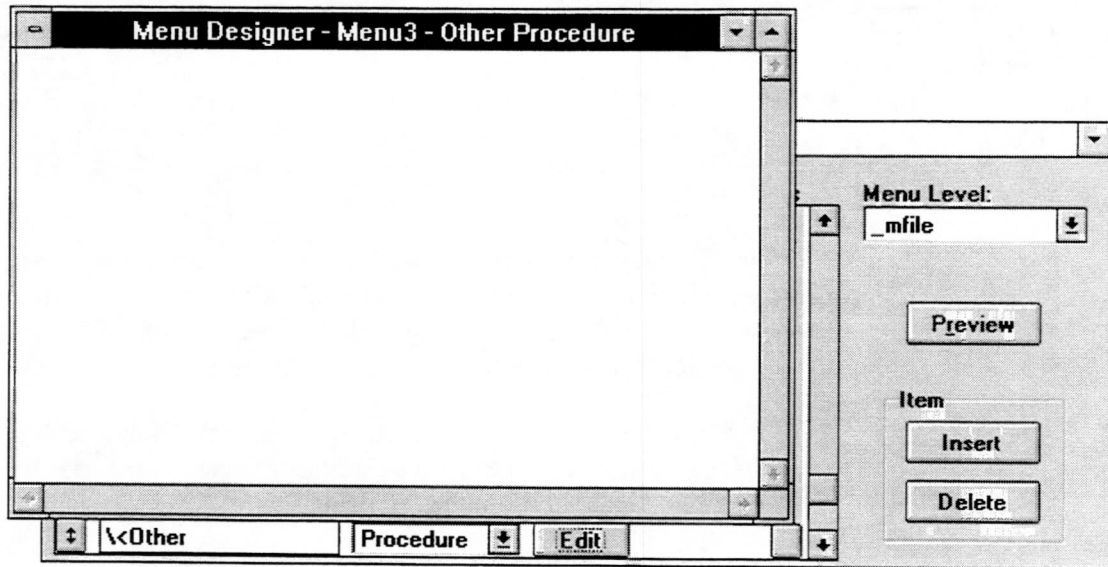


You may also enter your own bar # (or a pad name at the menu bar level). If you use your own bar #, be sure to include a command like the following somewhere in the menu cleanup procedure code:

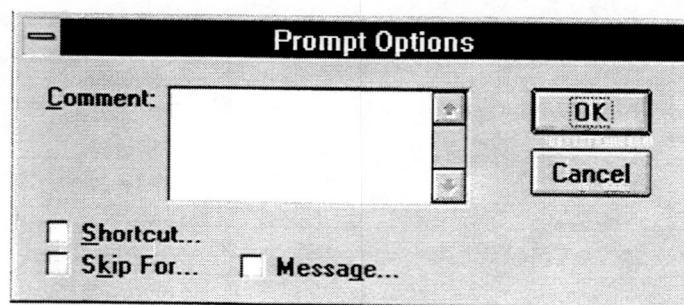
ON SELECTION BAR <your barnumber> of <popupname> <command>

This command defines the action that should occur when you select your bar #.

The *Procedure* result, or *Proc.*, places a Create or Edit push button to the right of the result field. Selecting this button opens a program editing window. A code snippet you write in this window is executed when the menu option is selected.



There are options that you can apply to each menu popup. To access these options through the menu options dialog box, select the Options check box to the right of the result field.

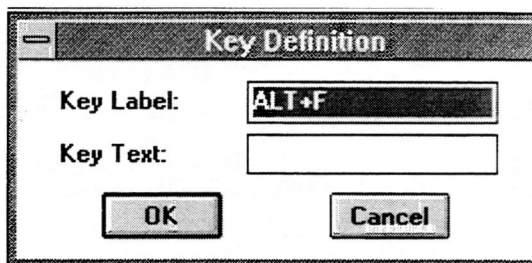


Enter any notes or comments about the menu option in the Comment text box. These are stored in a memo field, but are not included in the generated source code.

The *Shortcut* option allows keys or key combinations to be attached to a specific menu option. When you select the Shortcut check box, the Shortcut

dialog box appears. Enter a key label definition and a key text definition in this box.

The *key label* definition must be a key label name used with the ON KEY LABEL command. The *key text* definition allows you to enter the hot key description that displays next to the relevant menu popup prompt. For example, to use Ctrl+A as a hot key, the key label definition is **Ctrl+A** and the key text definition could be ^A. Hot keys are active whenever the menu option is enabled.



NOTE: Hot keys for menu bar pads are handled differently than menu option hot keys. The menu bar shortcut key is always the ALT key plus whatever letter you prefixed with the “\<” characters in the pad prompt.

WARNING! Always use key combinations when assigning menu option shortcut or hot keys. Use Alt or Ctrl plus some character. Do *not* use single characters since they would conflict with keyboard input in your application.

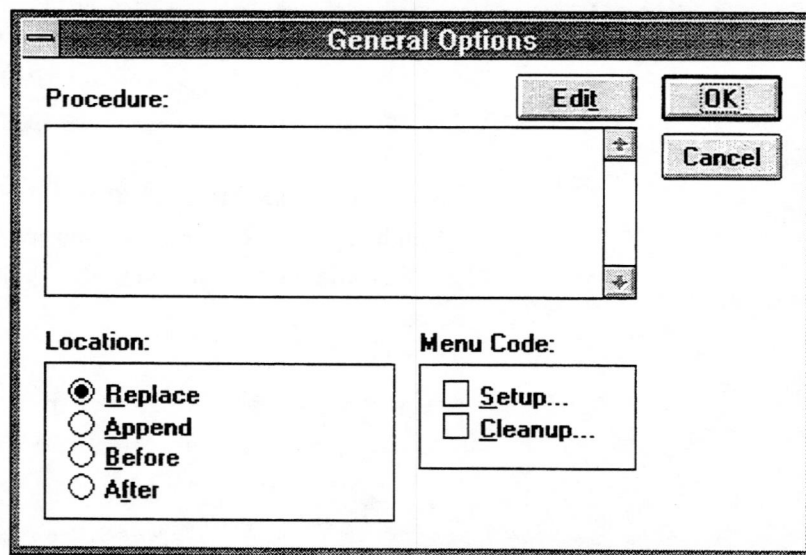
With the option called *Skip for*, you enable or disable a menu pad or popup based on some logical condition that you define. When the logical condition is .T., the menu pad or popup is disabled. When it is .F., the menu pad or popup is enabled. Selecting the Skip for option displays the FoxPro Expression Builder dialog box. Simply enter the expression and click OK. The Skip for option is handy and quite powerful for controlling menu option disabling and enabling.

When you activate the FoxPro Menu Designer, it adds two menu bars to the View menu and creates a new Menu pad. There are five options available from the MD Menu pad. When you create a new menu, the Quick Menu option is enabled. This option is only available when the menu design window is empty. (See the previous section for more information about this option.)

The Insert and Delete options allow you to insert and delete menu option lines. These options are also found on the Menu Design and Submenu Design windows.

To bring up the *General Options* dialog box, select View-General Options. From this dialog box, you can do the following:

- Create setup and cleanup code.
- Create a general procedure, a routine that executes for options where no action is specified.
- Specify a global mark character.
- Specify the location of menu pads on the menu bar.



Selecting the Setup check box brings up an edit window in the background. Click OK to access this window. GENMENU generates the setup code and places it before the menu definition code. Unlike the case with screens, most developers prefer to leave the menu setup code empty. You can move application startup logic to the menu setup code area, but it is easier to maintain in the calling PRG.

When you select the Cleanup check box, an edit window also comes up in the background. Again, click OK to access the edit window. GENMENU places the cleanup code *after* the setup and menu definition code, but *before* any code that assigns actions to menu pads or menu options. This is an ideal place to set the initial enable and/or disable status of menu pads or bars, or to preactivate an initial menu option with a command like the following:

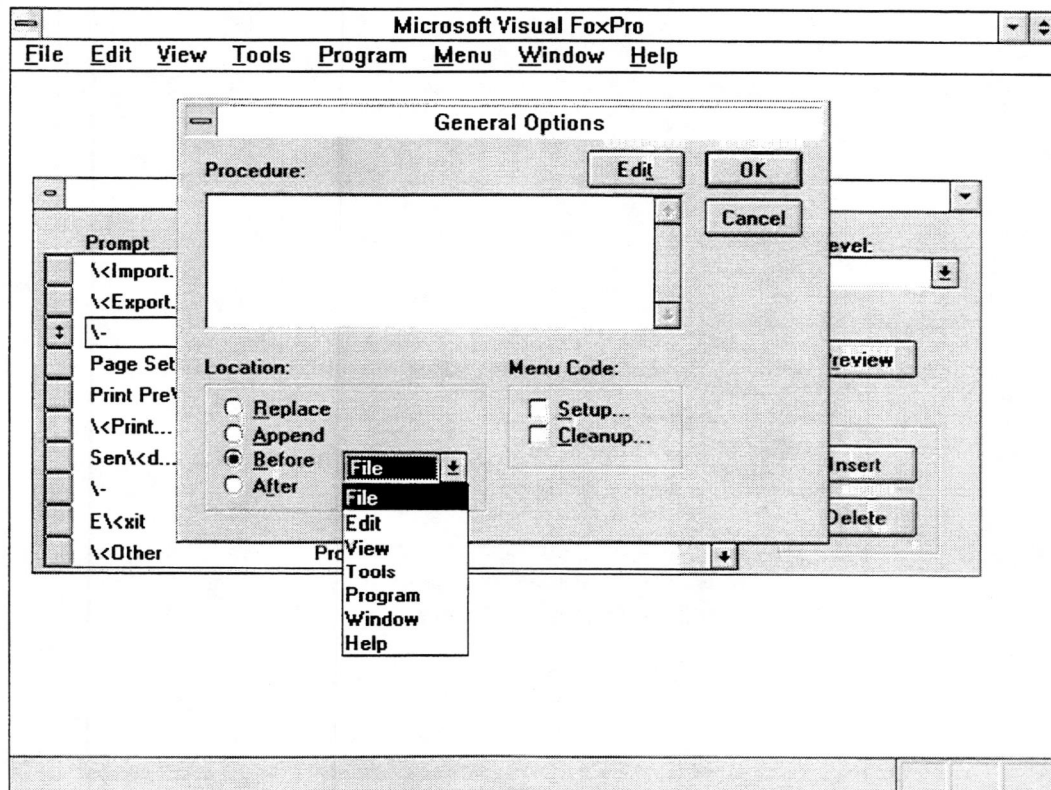
```
ACTIVATE POPUP edit BAR 3
```

The Procedure edit *box* enables you to create a procedure that executes for each menu option selected. This is provided where the option has no local procedure defined. This feature is good for centralizing the selective enabling and disabling of menu options or for displaying certain messages to the user based on the selected menu option. Select Edit to permit editing of this edit box. Then click OK to activate the box.

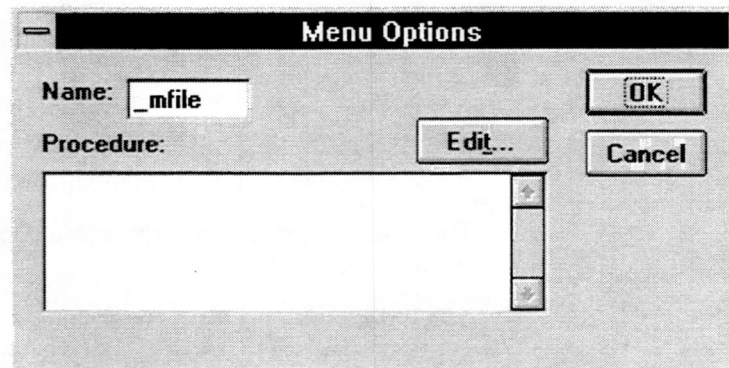
The following Location options let you control how the MD integrates your menu into the system menu bar:

- The Replace option is the default, and causes your defined menu to replace the currently defined menu.
- The Append option adds your menu pads to the end of the current menu (on the right).
- The Before option allows you to place your menu pads before an existing menu pad. When you select Before, a popup list of menu pads that are available for your selection displays.
- The After option works just like the Before option, except that it places your menu after the selected menu pad.

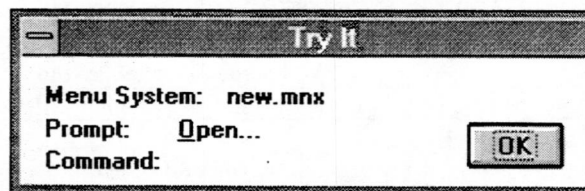
Note that the Before and After options assume that the available bars on the existing menu are the same at runtime as they are at design time. For all practical purposes, this makes them useful only for modifying the default FoxPro system menu bar.



When you select the Menu Bar Options, the Menu Options dialog box displays. The only option available in this dialog box is the Procedure edit box. Select Edit to bring up the edit box and click OK to activate it. The procedure that you enter in this box executes for every menu option at the current menu level that has no local procedure defined.



Probably the most useful button in MD is the Preview button. This button allows you to actually test the look and feel of your defined menu without generating any code and without actually running the program (MPR). Use this time-saver whenever possible. When you select this option, the Try It dialog box appears and displays the prompt of any menu option you select. It also displays the procedure or command that is executed if one exists.



From a strategic point of view, you really only have a few reasonable choices for implementing your menus into an application:

- Design and build a complete menu, completely replacing FoxPro's standard system menu. This is a good choice if you are comfortable with the MD and have specific menus you use for all applications.
- Create a standalone menu if you are adding just one menu pad to an existing menu definition. For example, you can execute NEWPAD.MPR to add your pad to the existing menu.
- Hand code the menu definitions. This is the least attractive choice. Avoid hand-coding the relatively complex syntax of menu creation commands whenever you can.

NOTE: There are a few limited circumstances where some form of hand coding may be required to supplement the Menu Designer. For example, to create a submenu based on a user-definable set of options, you may need to create a routine to parse user-input in a text file or FoxPro table. Then you need to create a menu popup or branch of popups from this input. But even in this case, you should use the Menu Designer to create the basic menu.

Creating a Startup Program

After creating a menu for your application, the final step in putting your application together is to create a startup program to launch your application. This startup program might have the responsibility of opening files, specifying global settings, and setting up the environment for your applications. The code below shows what a startup program commonly looks like:

```
*-- Set global options
CLOSE ALL
CLEAR ALL
SET EXCLUSIVE OFF
SET TALK OFF

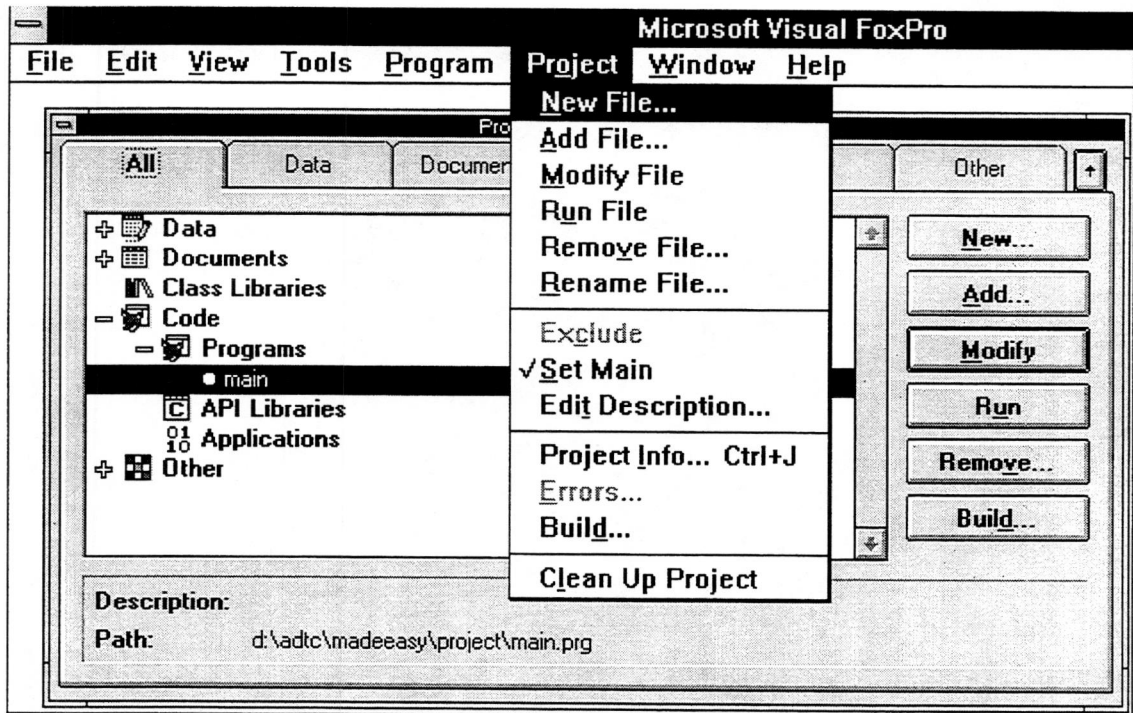
*-- Set system titles
_screen.Caption = "My New Application"

*-- Load menu
DO MAINMENU.MPR

*-- Set wait state for your application
READ EVENTS

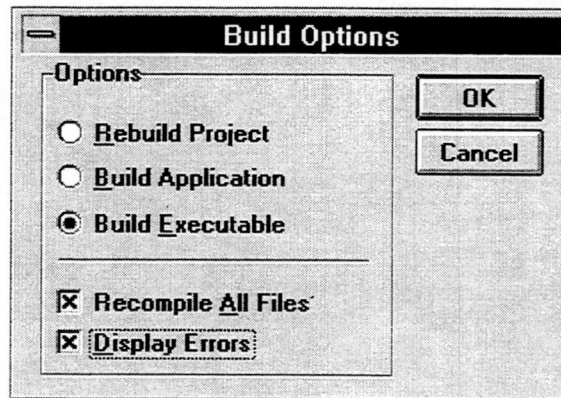
*-- Clear environment and QUIT
SET SYSMENU TO DEFAULT
CLEAR ALL
CLOSE ALL
QUIT
```

After creating a main program you need to specify it as the program that gets launched upon running your application. You do this by selecting your startup program in the Project manager and selecting Set Main from the project menu (see below). Selecting this option puts a bullet character next to the program that will be launched whenever your application is run.



Building Executables

The last steps in creating your application for distribution is to create an Executable file. Creating an executable is remarkably simple. To create an executable you simply open your project and select the Build button. The figure below shows the Build Options dialog from the Project Manager. Choose Build from the Project menu to get here. When you own the Professional version of Visual FoxPro the Build Executable option is available.



All you have to do is select the Build Executable option and click OK. The project builder then builds an EXE file that you can distribute. Another method of creating executables is to use the BUILD EXE command. The syntax of this command is:

```
BUILD EXE contact.exe FROM contact.pjx
```

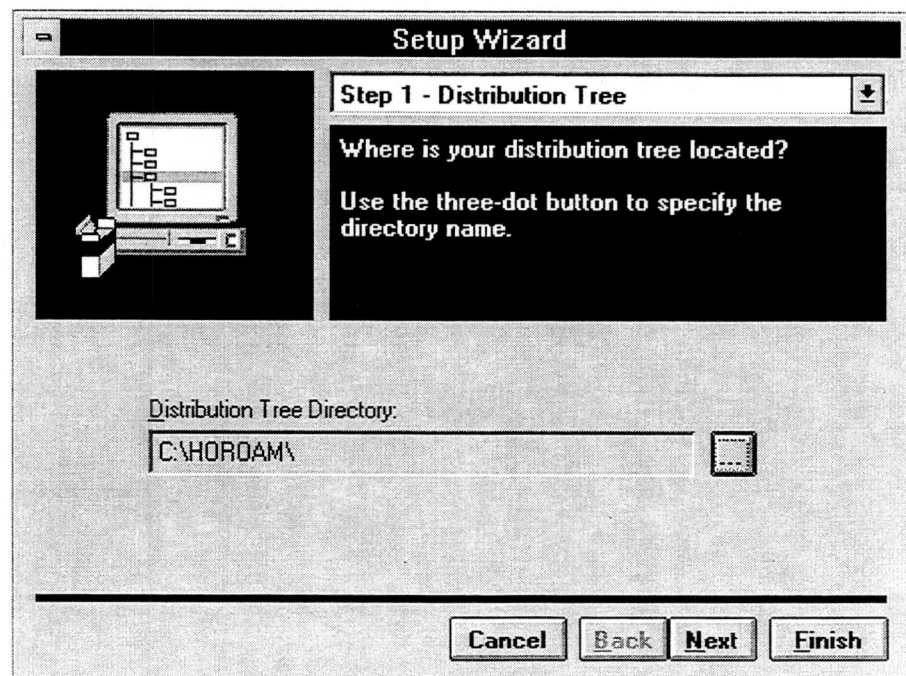
Building Setup Disks

After you create an executable file, you need to use the Visual FoxPro Setup Wizard prior to sending your application out into production. The Setup Wizard is an application that processes your application files and creates a set of disk images ready for distribution. In order to distribute your applications, you **MUST** run the Setup Wizard to create a set of disks. This is due to the number of external libraries that Visual FoxPro uses to create applications. You can no longer just send out your .EXE, your data, the .ESL library files, and expect Visual FoxPro to work. Visual FoxPro 3.0 implements ODBC, OLE, and WIN32S (under Windows 3.11) in its architecture. It is FoxPro's use of these Windows components that requires you to use the Setup Wizard.

The Setup Wizard shows a standard Wizard style of screens in order to create a group of setup disks. The steps that the Setup Wizard goes through are documented below.

Screen 1 - Distribution tree

The following screen represents the opening screen of the Visual FoxPro Setup Wizard. This screen requires you to specify the path of the application you want to create distribution disks. Upon specifying the path for your application the wizard prompts you with six more screens allowing you to specify installation parameters. These screens are discussed below:



Screen 2 - Components to install

This form allows you to specify the operating system components to install and which operating systems to install. You can specify parameters for:

- Windows 32s (Required for applications running Windows 3.x)
- Windows 95 (Installs required files for Windows 95)
- Windows NT (Installs required files for Windows 95)
- Visual FoxPro runtime library (necessary for Visual FoxPro .EXEs)
- ODBC (Necessary for applications that use remote views)
- MS-Graph (Necessary for applications using graphs)

Screen 3 - Disk images to create

This form allows you to specify the type of installation disks to create and where to put the disk images. It is recommended that you keep your installation disks in a directory separate from your application. You can create disk images for:

- 1.44 mb floppies
- 1.2 mb floppies
- Network installations.

Screen 4 - Setup and copyright information

This form allows you to specify copyright and installation title information that will be displayed when the users run your setup program.

Screen 5 - Program Manager information

This form allows you to specify whether users are allowed to change directory and program manager information during the installation process.

Screen 6 - File Summary information

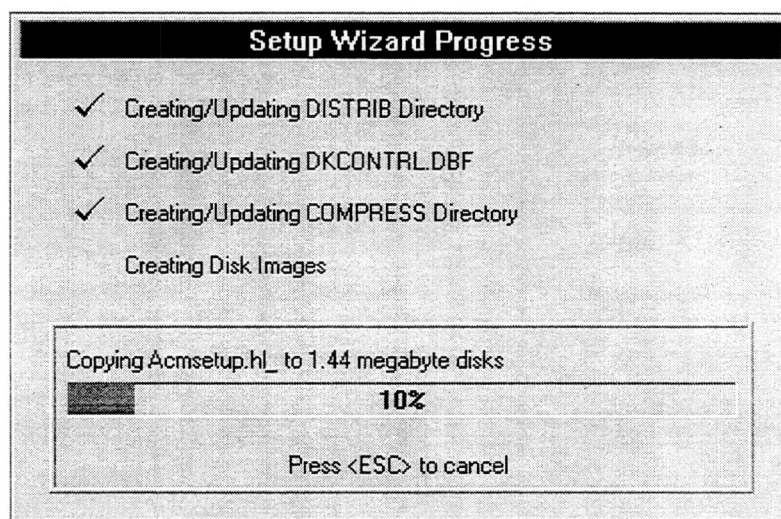
This form allows you to specify whether a file to be installed should be placed in the program group created by the installation program. You can also specify whether the selected file is an OLE component that needs to be registered with windows.

NOTE: If you are installing any OCX controls, remember to copy the OCX controls into your distribution directory and check the OLE option for the file in screen 6.

NOTE: If you make any 16 bit .DLL calls using FOXTOOLS.FLL you will need to copy DDEREG.EXE from the VFP directory into your distribution directory. Visual FoxPro uses DDE to make API calls requiring you to install this file.

Screen 7 - Finish line

This form completes the Setup Wizard options and allows you to create installation disks. After you finish the Setup Wizard options, Visual FoxPro creates your distribution disks. The following illustration represents the Setup Wizard progress dialog:



After completing your distribution disks, the Setup Wizard prompts you to view statistics of the installation process and print a report showing which files the Setup Wizard placed on which disk.

Disk	Files	Bytes Used	Bytes Available	% Used
1	26	1457664	0	100.0
2	8	1457664	0	100.0
3	8	1457664	0	100.0
4	27	1457664	0	100.0
5	80	97280	1360384	6.67

Finally, if you do not like how some function of the Setup Wizard works, you can alter the source code yourself because it is provided with the Visual FoxPro Professional Edition.

Using OLE In Visual FoxPro

Objectives

Now that you understand how to create the components of a Visual FoxPro application lets take a look at something really fun. Visual FoxPro now supports OLE 2.0. This section looks at the components of the OLE specification provided in Visual FoxPro.

In this section you will learn:

- What OLE is.
- The different features of OLE.
- How to add OLE objects to your forms.
- What OLE automation is.
- How to use OLE custom controls in your forms.

Understanding OLE

From time to time you might write a Visual FoxPro application that needs to communicate with other Windows applications. The modern way to communicate between Windows applications is through a Microsoft technology called OLE (pronounced “oh-lay”).

Why Use OLE ?

OLE simplifies communication between Windows applications. Microsoft embraces OLE fully in their applications and encourages you to do the same in yours. You can develop Visual FoxPro applications that have the ability to link with other applications. By using OLE custom controls, you are not limited to the FoxPro development environment. If you want to add imaging or a set of scroll bar controls to your application, go buy them. If you want a better graphing package, go buy one and call it from your Visual FoxPro applications. OLE and Visual FoxPro are a powerful combination.

History Of Inter-Application Communications In Windows

No discussion of OLE would be complete without a discussion of the history of Windows communication mechanisms. The Windows clipboard is one of the most frequently used ways to exchange data in Windows. You can easily copy data to the clipboard and then into other parts of the same, or other documents. Windows users commonly use the clipboard to copy static information from one document to another. This information is not linked between the two documents and is thus "static".

If you were to place the evolution of Windows inter-application communications on a timeline, it would look like this:

Clipboard ----> DDE ----> OLE 1.0 ----> OLE 2.0 -----> ??

(past)

(present)

When Windows was first introduced, the clipboard seemed like a revolutionary idea. Although it seems rather tame by today's standards, the clipboard continues to be useful.

DDE is a communication mechanism that allows applications to exchange both data and commands. DDE provides the ability to link data from unrelated applications.

OLE 1.0 adds a layer of sophistication to DDE by providing a mechanism for encapsulating DDE functionality into visual objects. DDE is primarily used by developers, whereas OLE is oriented to the end-user. It is much easier for you to initiate an OLE link than it is for the end-user to establish a DDE conversation directly.

Visual FoxPro supports version 2.0 of the OLE standard. OLE 2.0 combines the best of DDE and OLE 1.0. One new OLE 2.0 feature is called *OLE Custom Controls*. *OCXs*, files containing a collection of OLE custom controls, can be created by component vendors for use in any OLE 2.0-compliant application, including Visual FoxPro. You can use OLE custom controls on forms in the same way that you use controls derived from a native Visual FoxPro control class or subclass. Another important new feature is *OLE Automation*. This allows you to use the functionality of another application within Visual FoxPro. For example, you could embed a Word document in a Visual FoxPro form. When you move to the Word object, the system menu changes to provide Word for Windows support.

OLE Defined

The newest member of the Windows communication family is OLE 2.0. OLE allows you to embed foreign file formats within the Visual FoxPro application. These foreign files include word processing files, spreadsheet files, and graphic images. OLE can do two things with objects: it can either link objects (similar to DDE) or it can embed objects.

Linking

When Visual FoxPro creates a link with another object, it simply stores the file name and some other internal OLE information within a general field. This tells Visual FoxPro where to access this information on the disk, as well as how to modify it. Linking is useful if you need to use other applications to modify the linked data. For example, you can use Excel to update a linked spreadsheet and see the changes from within your Visual FoxPro application.

Embedding

OLE can also embed objects. Visual FoxPro's embedding mechanism occurs within a General field. Rather than storing only text or numbers, you can store entire spreadsheets and word processing documents as objects within your database files. You can use OLE 2.0 to create and control access to databases of legal documents with your Visual FoxPro application. Or, you can create collections of graphics to display for art collectors. The possibilities are endless.

Object Registration

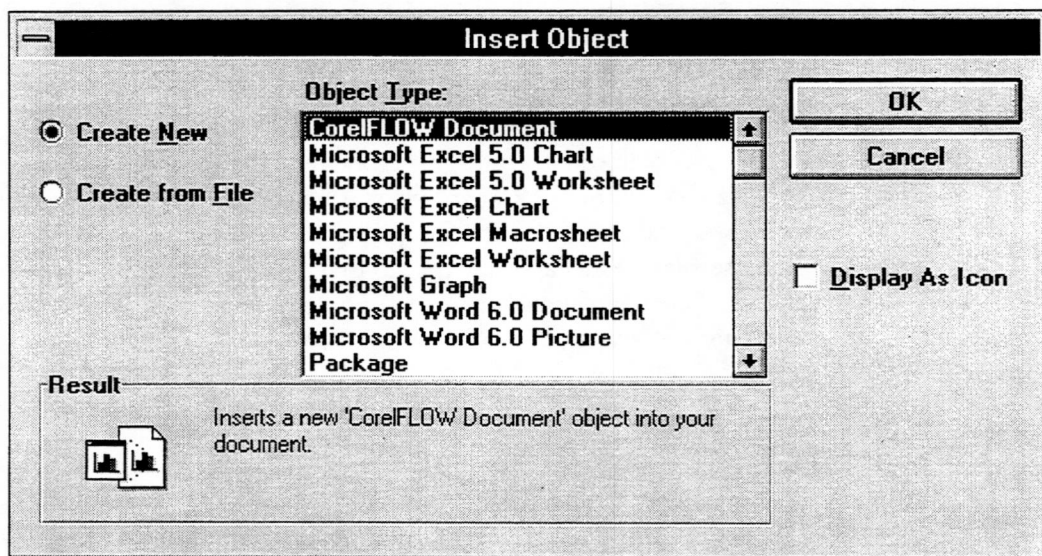
Applications support OLE in different ways, either as a client, as a server, or as both. OLE servers generally register themselves with Windows. The registration process determines how each OLE server handles its respective objects. Registration determines what program your server calls when it needs to update a file, how you edit an object, how your application should display the object, etc. Windows stores all registrations in a file called REG.DAT. You can view or modify this file with a program that comes with Windows called REGEDIT. With REGEDIT, you can determine which servers are available to your system.

Client Operation

Visual FoxPro supports OLE as a client application. This means that Visual FoxPro can embed OLE objects within its data structures. Visual FoxPro's OLE client mechanism exists within the General data type. This General data type is where to focus the bulk of your OLE operations.

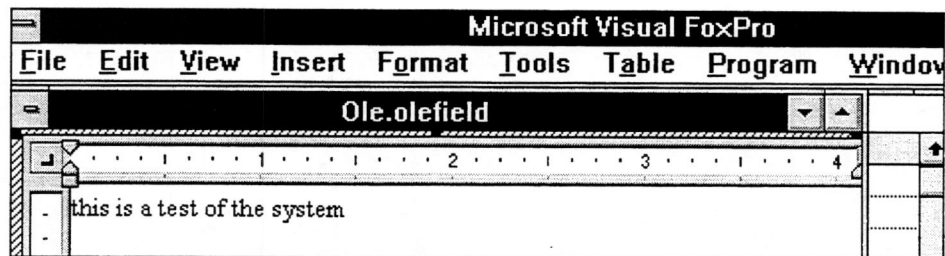
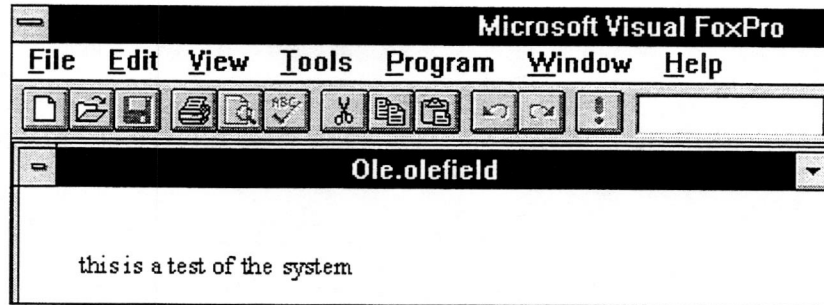
Creating and Receiving Objects

The simplest way to use OLE is to create a database that contains a General field within its structure. Then, append records to the database and edit the General field. To enter information into the General field, use the Insert Object option from the Edit menu. This option brings up a dialog box with a list of the registered OLE object types. Choose one of these object types, and then Visual FoxPro activates that particular OLE server and allows you to create an embedded object. Below is an example of the dialog box:



The Create From File radio button allows you to insert a file located on disk into your Visual FoxPro table. Use this dialog box to load previously scanned images, word processing documents, or spreadsheets into a General field in a Visual FoxPro table.

When you insert an object into your database, the Visual FoxPro menu changes. The two illustrations below demonstrate the changes that occur when you insert a Word document into a General field in a table.



Note the menus. Whenever you edit an OLE object, the menu changes to that object's server menus. This is known as **in place editing**.

You can also create OLE objects by using the APPEND GENERAL command. APPEND GENERAL allows you to insert objects into general fields. The syntax for APPEND GENERAL is as follows:

```
APPEND GENERAL ole.oleobj FROM C:\WINDOWS\ZIPZAG.BMP
```

This syntax embeds ZIPZAG.BMP into a field called oleobj. Use the LINK keyword with the APPEND GENERAL command to create linked files instead of embedded files. This is useful if applications other than Visual FoxPro need access to the linked file.

Manipulating Objects

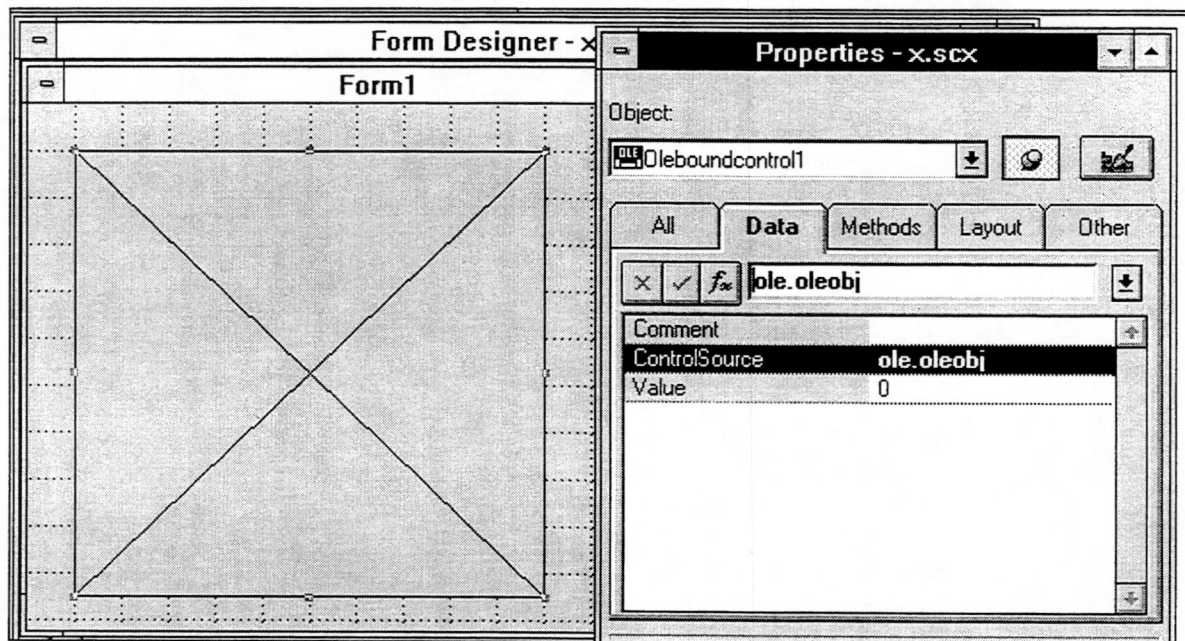
Visual FoxPro provides an array of excellent facilities to manipulate OLE objects. From within your Visual FoxPro applications, you can display objects on the screen, print objects on reports, modify objects, and use the facilities of OLE servers using the OLE automation language.

Using OLE Objects In Forms

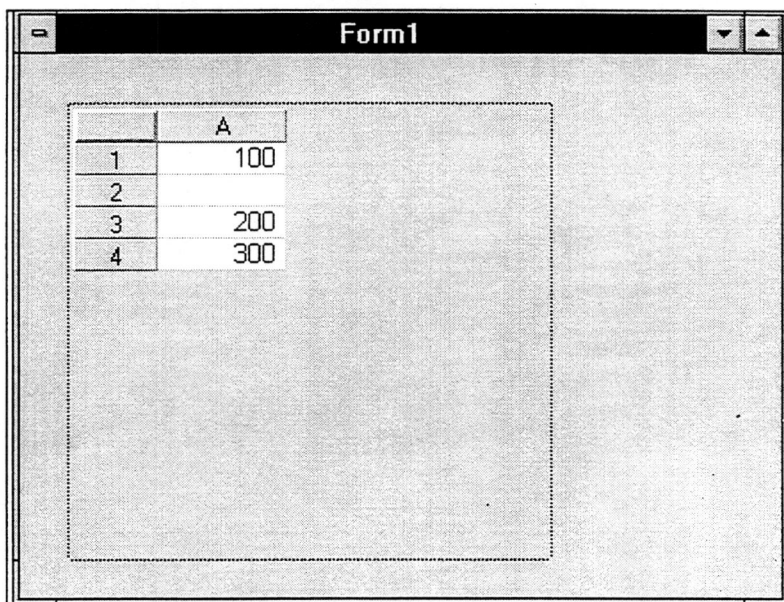
You can embed OLE objects in your databases and you can edit them in your Visual FoxPro forms. There are two ways to add OLE objects to your forms in Visual FoxPro. You can use an OLE Bound Control or an OLE Container. The differences between the two are described below.

OLE Bound Control Objects

OLE bound objects allow developers to display and edit OLE objects contained within a General field of a Visual FoxPro table. The following illustration shows you what an OLE bound control looks like at design time:



Bound controls use the ControlSource property to specify which General field to use as the data source. The illustration below represents the same form at runtime:



	A
1	100
2	
3	200
4	300

OLE Container Objects

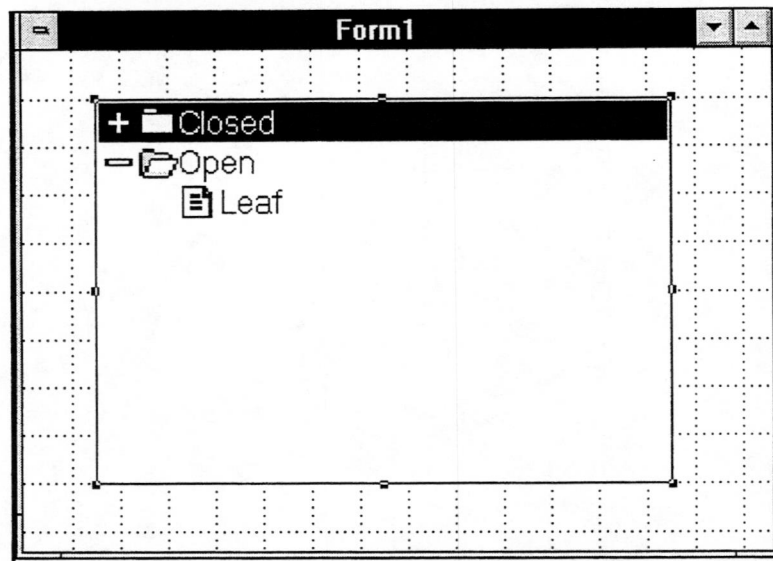
OLE bound controls are linked to a General field found in a table. OLE containers have no such link. OLE containers can contain template OLE objects, links to files, and OLE custom controls. OLE custom controls form designer widgets used to extend the functionality of Visual FoxPro. Developers commonly use OLE containers for OLE custom controls or for representing an OLE object that is not saved to a Visual FoxPro table. For example, a spreadsheet object which is not saved to a FoxPro table is one example of unbound data.

Using OLE Custom Controls

One of the most significant enhancements to Visual FoxPro is the ability to use OLE custom controls contained in OCX files. When Microsoft introduced Visual Basic in the early 90's they created a new file type known as a VBX, a collection of Visual Basic custom controls. This file type created a market of reusable components that could be inserted into applications with relative ease. With the growth of the VBX third-party market, Microsoft began looking towards the future of application development and created the OCX architecture.

OLE custom controls add functionality that Visual FoxPro does not provide in its native development environment. You can buy reusable components to

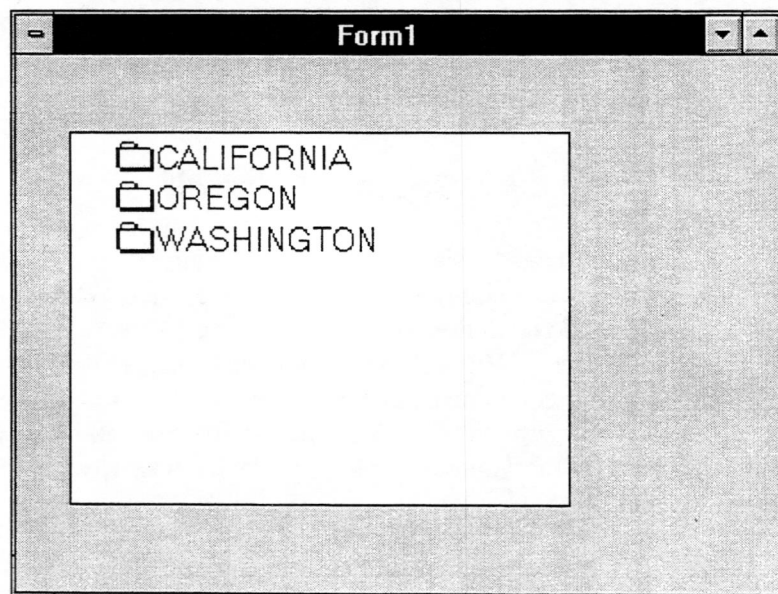
insert into your applications. The examples below demonstrate a few OLE custom controls that you can use in your Visual FoxPro applications.



The above illustration shows the *outline custom control* provided with Visual FoxPro. You can add your own outline headings and subheadings to this outline at runtime by placing the following code into the Activate method of your form:

```
THISFORM.OLECONTROL1.ADDITEM("CALIFORNIA")  
THISFORM.OLECONTROL1.ADDITEM("OREGON")  
THISFORM.OLECONTROL1.ADDITEM("WASHINGTON")
```

Now your form looks like this:



Visual FoxPro includes the following OLE controls in the Professional Edition:

- MSOUTL32.OCX - Outline control
- MSCOMM32.OCX - Serial communications control
- MSMAPI32.OCX - E-mail control
- PICCLP32.OCX- Imaging custom control

WARNING! The only control that you can use in the Windows 3.1 environment is MSOUTL32.OCX. The other controls are 32-bit and require Windows 95 or Windows NT. *← IS THIS CORRECT, MARTY*

Non-visible OLE Custom Controls

Not all OLE custom controls have visual characteristics. Visual FoxPro provides two OLE controls that have no visual characteristics: the MAPI (Mail Message and Mail Session) and MSCOMM (communication) controls. These controls can be placed anywhere on the form that needs to access the control. Although the OLE custom control icon is visible at design time, it will be invisible to the user during run-time. With Visual FoxPro's object-oriented capabilities, you can also create your own custom objects using OLE custom controls as the template.

Other OLE Features

OLE Automation

One of the newest features of OLE 2.0 is that it can control other OLE servers via the Automation interface. Although not all applications support OLE Automation, the feature is already present in most of the applications found in Microsoft Office. Visual FoxPro uses OLE Automation to control a server application through the use of a set of exposed methods and properties. This mechanism allows you to transfer data and commands between OLE applications. The following code demonstrates how OLE Automation works.

```
oleApp = CREATEOBJECT("Excel.Application")
oleApp.Visible=.T.
oleApp.Workbooks.Add
oleApp.Cells(1,1).Value=7
```

```
oleApp.ActiveWorkbook.SaveAs ("C:\TEMP.XLS")  
oleApp.Quit
```

The above code creates a new Excel spreadsheet, puts a value in cell 1,1 and saves it as TEMP.XLS.

<p>Tip: Most OLE Automation servers use their respective macro languages as their automation languages. To learn how to program an automation server, record a macro and mimic the results in your application.</p>
--

The Internet Wizard

Introduction

This spring Visual FoxPro will have an addition to its already cool set of wizards: The Internet Wizard. The Internet Wizard assists Visual FoxPro developers in creating HTML pages that can access Visual FoxPro databases. This section highlights some of the features of this wizard.

In this section you:

- Learn what the Internet Wizard does.
- Use the Search Page Wizard
- Learn how Visual FoxPro manages Internet access

What the Internet Wizard Does

The Internet Wizard functions as a tool that creates a basic framework for transmitting data requests to a Visual FoxPro data server, and receiving those answers back. The Internet Wizard consists of three main components:

1. The Search Page Wizard: A wizard that creates an HTML page that allows users to specify query information to be passed to the Visual FoxPro data server.
2. VFPCGI.EXE: A program that passes a CGI script to the Microsoft Internet Server, which hands the query to the Visual FoxPro data server.
3. The Visual FoxPro WWW Data Server: An application that polls for a specific set of files, processes the queries found in those files, and returns an HTML Web Page to the requesting party.

These components work in conjunction with a Web Browser, such as Microsoft's Internet Explorer or Netscape's Navigator, to create queries and retrieve the results from an Internet server, such as Microsoft's Internet Information Server. In order to utilize the Visual FoxPro Internet Server you must have the following:

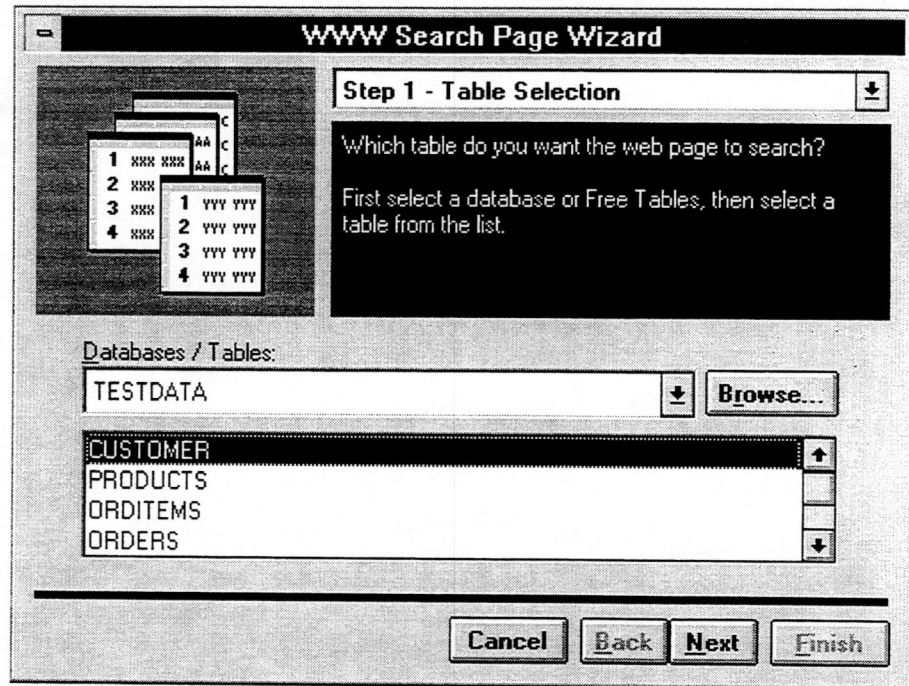
- Visual FoxPro 3.0b
- An Internet server program
- An Internet viewer

Once you have acquired and installed these components, you can begin creating your Web pages.

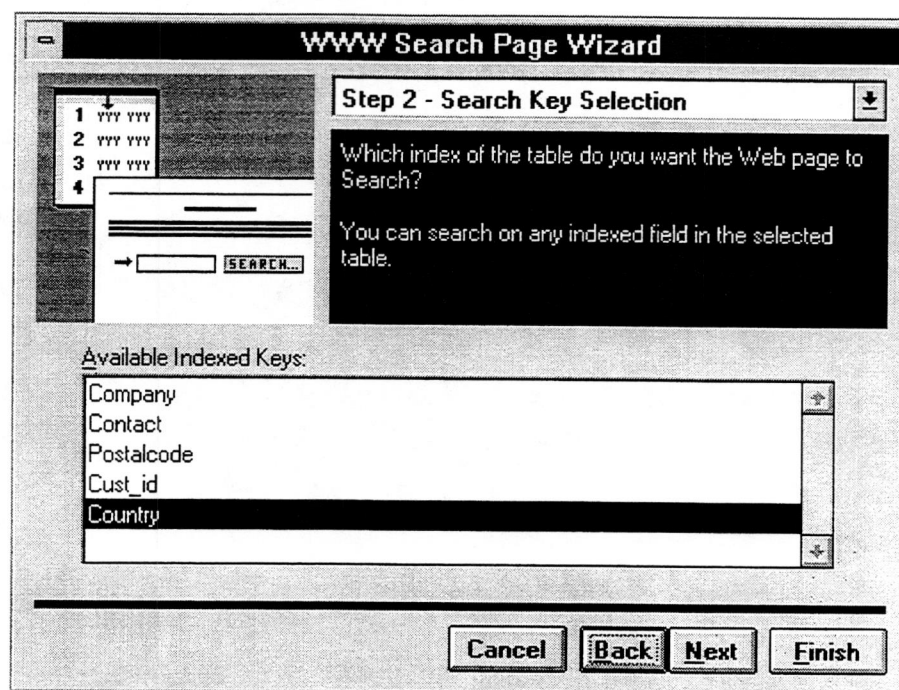
Using the Search Page Wizard

The Search Page Wizard is a tool that can be used to generate an HTML page that can be accessed by users to retrieve data from your Web Site. The Search Page Wizard functions much like other Wizards. It takes you through a set of predefined steps that ultimately result in a set of files that comprise your Web Page and its components.

The first step of the Search Page Wizard is to select the table you want to select data from. The following screen shows the table selection form:



Upon selecting the table you want to query, the wizard prompts you to select an index to be used as the selection criteria for your web page. The queries that the Wizard generates are limited to the indexes found on the table you selected. The following screen shows the Search Key Criteria page of the Internet Wizard.



The next set of steps deal with the appearance of the web page to be generated. They allow you to specify:

- The Title and Description of your web page.
- A header and background image.
- Whether the answer file can be saved as a file.

The following illustrations show the appearance selection screens:

WWW Search Page Wizard

Step 3 - Search Page Title and Description

What text do you want displayed on the web search page?

Enter text for the title and description of the web search page. The description will appear just below the title.

Search Page Title:
ADTC Internet Wizard Demonstration

Search Page Description:
This Web Page allows you to query customer information by country.

Cancel Back Next Finish

WWW Search Page Wizard

Step 4 - Search Page Setup

Do you want to include a background or header image on the web search page?

You may select any JPEG or GIF file.
You can also provide an option to return the search results as a file.

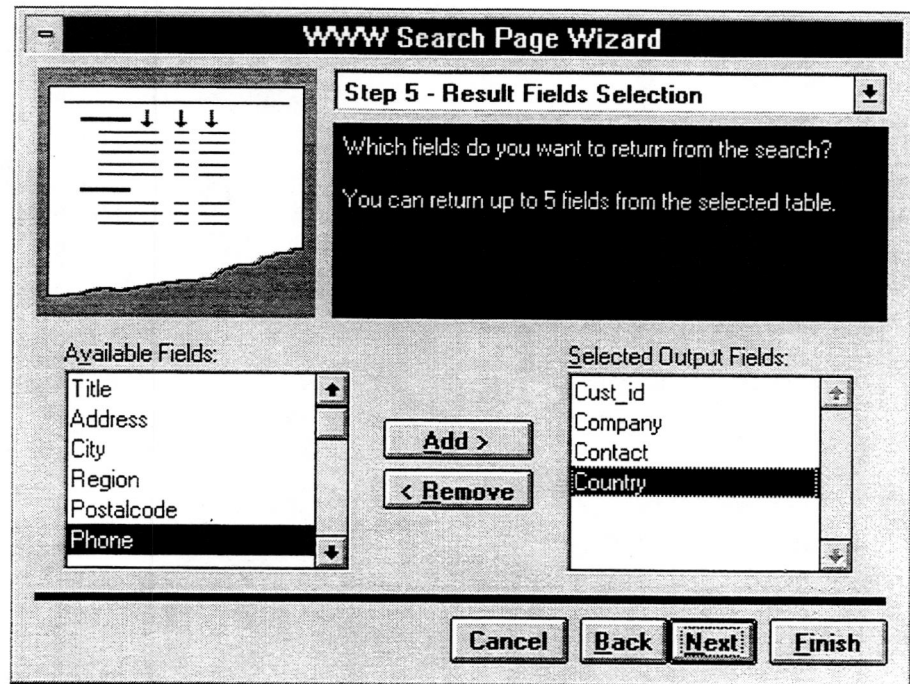
Background Image...

Header Image...

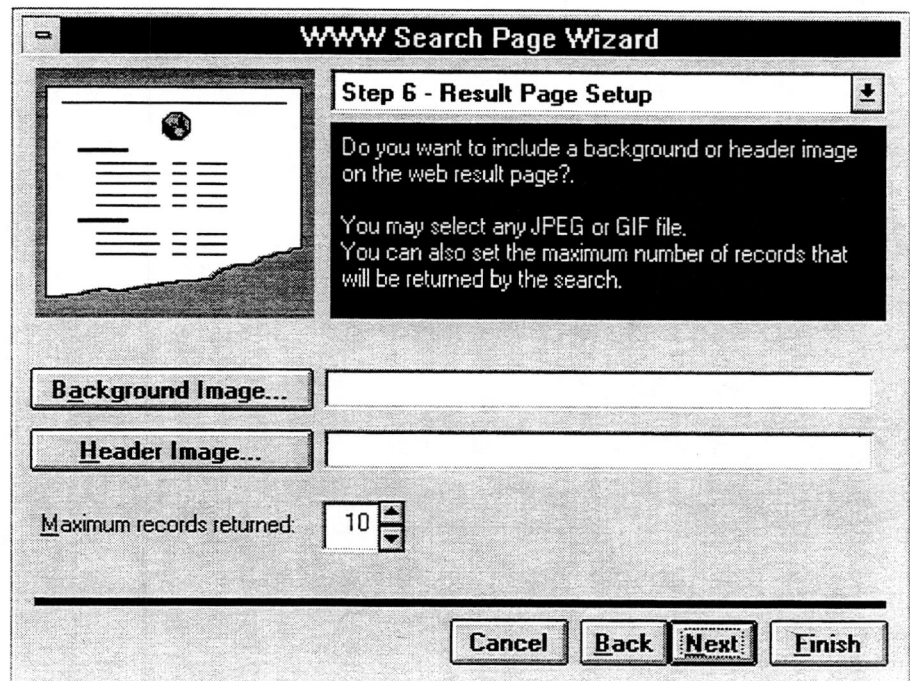
☒ Provide the ability to download the result set as file

Cancel Back Next Finish

The next step is to select the fields that will be returned by the Visual FoxPro Data server. You are limited to returning up to five fields. The following screen shows the field selection dialog:



Finally, you can specify the appearance of the returned web page. You can specify a background and header image and also the number of rows to return.



After completing your selections, you can generate your HTML page and supporting files. The Search Page Wizard generates the following items:

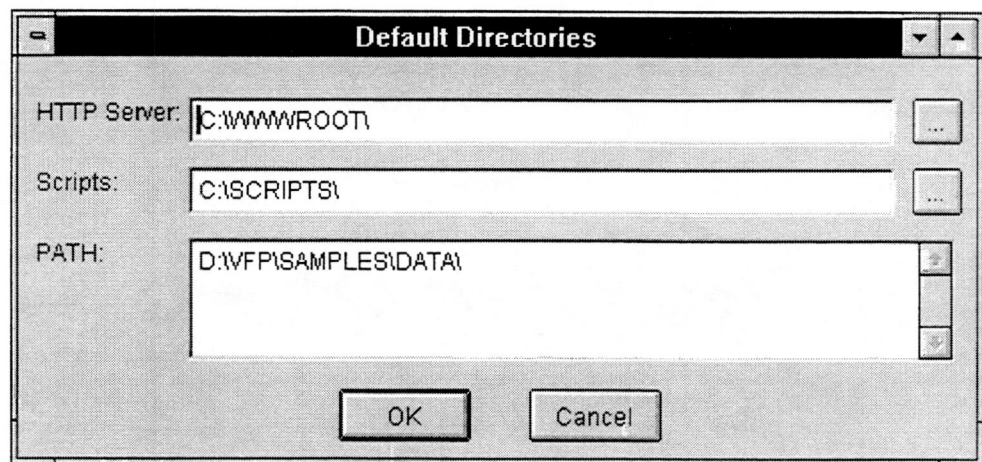
1. An HTML Page with the extension .HTM. This is your query page.
2. An HTML Page with the extension .HTX. This is the page that will be used to generate your return set.
3. A script file with the extension .IDC. This is the script for the CGI program.

After generating your web page components, you need copy them to the HTML and Scripts directories on your Internet server. Contact your Webmaster for more information about your specific installation.

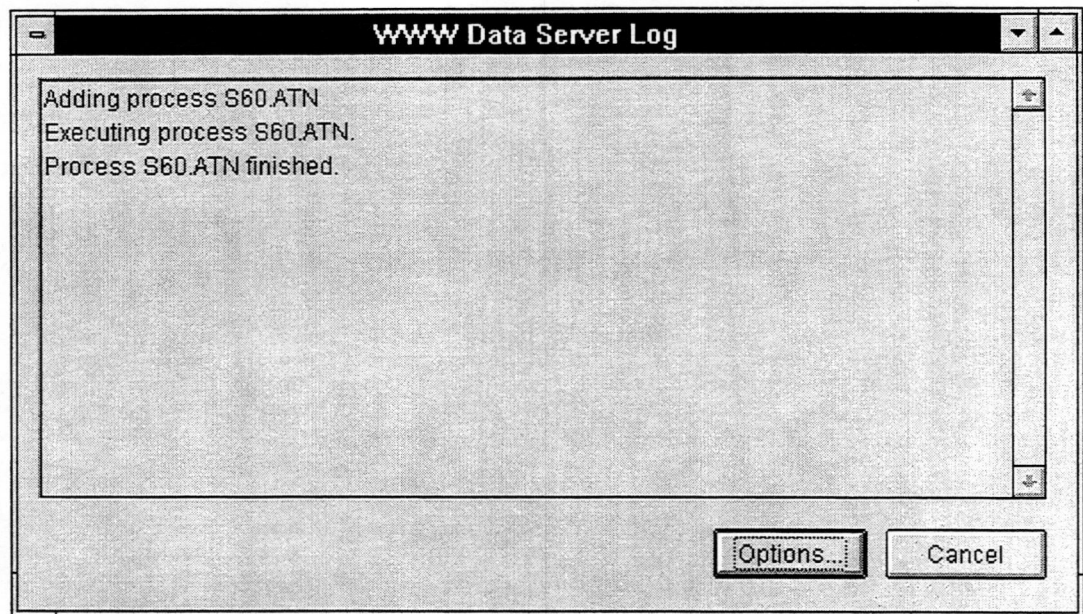
Running Your Web Server

After generating all of the web page components with the Search Page Wizard, you can begin processing data with your new Visual FoxPro data server. The first step is to run the Visual FoxPro data server. Do this by switching to the Command Window and running SERVER.APP.

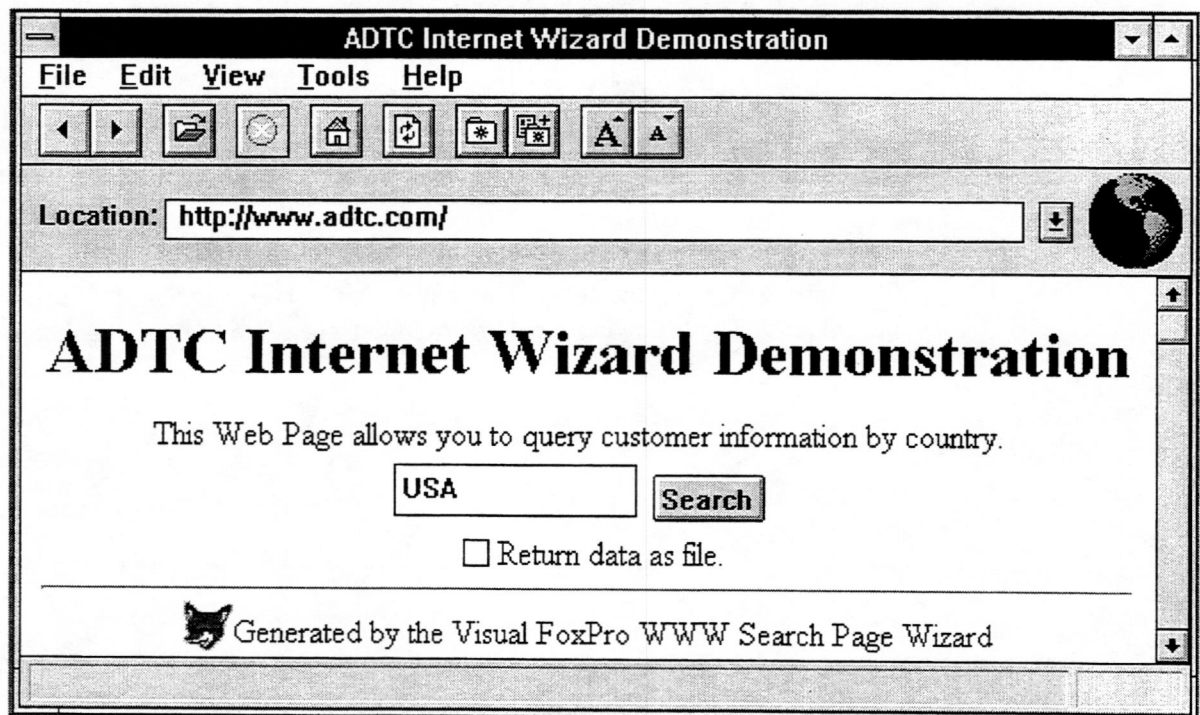
This is the program that polls waiting for server requests and returns the answers as an HTML page. Upon running the server for the first time you will be prompted for the directories of your web home pages, scripts, and path to your data. See the following illustration:



After specifying the appropriate paths, your server waits until it receives a data request. The following screen shows the Data Server processing data requests:

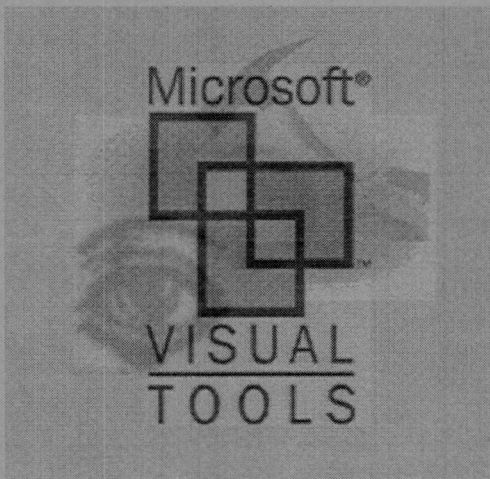


The following screen shows our generated web page in the Internet Explorer:



Just The Beginning...

The Visual FoxPro Internet Wizard is just the beginning foundation of Internet enabling your Visual FoxPro applications. When you look at the HTM/HTX and IDC files generated by the wizard, you find that they are simply HTML ASCII text files that can be altered to meet your needs. You can add more functionality to your Web Server as your knowledge grows. Also, Microsoft has provided you with the source code to the server application so you can alter the server to meet the specialized needs of your company. The possibilities are endless. . .



Microsoft® Developer
Seminar-in-a-Box Series

Visual FoxPro 3.0
Demonstration Script

Introduction

This demonstration script was developed for the *"Building Solutions with Microsoft Visual Tools"* Seminar. Files mentioned in this document can be found on the CD-ROM distributed with the seminar.

It is essential that the person doing the demonstration be familiar with Microsoft Visual FoxPro 3.0 and the development disciplines associated with creating Client/Server databases. Before doing this demonstration, be sure to read the associated Whitepapers located on the seminar CD-ROM.

Setup Instructions

This demonstration assumes that you are running either Microsoft Windows NT Workstation or Server 3.51, Microsoft Windows 95, or Microsoft Windows for Workgroups 3.11 and have installed Microsoft Visual FoxPro 3.0 Professional Edition.

System Requirements

Specification	Minimum Requirements
CPU	486 DX2/33MHz
RAM	16 M
Free Disk Space (for the demo files)	N/A
Operating System	Windows NT 3.51 (Workstation. or Server), Windows 95, or Windows for Workgroups 3.11
Additional Software	Microsoft Visual FoxPro 3.0 Professional Edition (complete install)

Note: Create a blank directory from within which to work.

New Design Tools

Feature	Narrative	Keystrokes
Project Manager	<p>Let's start by taking a look at the project manager in Visual FoxPro. I'll start by creating a project.</p> <p>As you can see, we now have a project window which contains a listing of all the elements of our project. From the "ALL" tab, all parts of our project are shown. We can also limit our project window to show only Data elements; our databases, tables, and queries.</p> <p>Or Documents; our forms and reports.</p> <p>Or any classes we may define.</p> <p>Then we have our FoxPro Code which consists of any FoxPro programs, API libraries, or FoxPro applications.</p> <p>And then we have the Other tab which contains our menus, text files, and other files like bitmaps, icons, etc.</p> <p>Another nice feature of the project manager is its ability to attach to our FoxPro menu and free up useful design space.</p> <p>But if at the moment I was just going to work with data, I could drop the Data window down.</p> <p>Or if I was going to work with the Data tab for some time, I could drag it off the menu bar and work with it independently.</p> <p>One of the great features of the Project Manager is in its flexibility to adapt to different programming styles.</p> <p>Another feature of the Project Manager is the ability to create a project and do all our development work from this tool.</p>	<p>Start Visual FoxPro</p> <p>In the Command Window, type "CREATE PROJECT MYPROJ"</p> <p>Click Data Tab.</p> <p>Click Document Tab.</p> <p>Click Classes Tab</p> <p>Click Code Tab</p> <p>Click Other Tab.</p> <p>Click and Drag project window up to menu bar.</p> <p>Click the Data button.</p> <p>Click and Drag the Data button off the menu bar.</p> <p>Close Data Window.</p> <p>Click and Drag Project Manager from menu bar.</p>

Feature	Narrative	Keystrokes
	<p>Let's take a look at some of the settings that are available to us in regards to our project. Here you can see that we can define an author, a company, a home directory. We can choose to include debug information, have the project encrypted, and attach an icon to our EXE.</p> <p>This gives us a list of all the files currently assigned to our project.</p> <p>Let's add a database to our project.</p> <p>I am going to use a database that ships with Visual FoxPro called TestData.</p> <p>As you can see, we now have the capability to drill down into our databases. Here you can see the name of our database container we just added.</p> <p>If I click it again, you can see a listing for tables, views, connections and stored procedures.</p> <p>Clicking on "tables" shows us a list of all tables available to us.</p> <p>If we click on any table, we get a listing of all fields available. This should give you a good idea of the level of information you can retrieve using the Project Manager.</p> <p>Now let's take a quick look back in our Project Info.</p> <p>Here you notice we now have a listing for our "testdata" database container.</p>	<p>Under the "Project" menu select "Project Info...".</p> <p>Click the Files Tab.</p> <p>Click on the Cancel Button.</p> <p>Click the Data Tab on the Project Manager, select "Databases" and click the button labeled "Add...".</p> <p>Use the Open dialog box to locate the file TESTDATA.DBC in the directory VFP\SAMPLES\DATA. (Instructor: This could be located elsewhere if you wanted it to.)</p> <p>Click the plus sign to the left of the word "Database" in the Project Manager.</p> <p>Click the plus sign to the left of the word "testdata".</p> <p>Click the plus sign to the left of the word "tables".</p> <p>Click the plus sign to the left of the word "customer".</p> <p>Click the minus sign to the left of the word "Database" to close the tree.</p> <p>Under the "Project" menu select "Project Info...".</p> <p>Click the Files Tab.</p> <p>Click the "Cancel" button.</p>

Feature	Narrative	Keystrokes
	<p>The Project Manager also serves the purpose of bringing together all our project's components into either a FoxPro application or a Windows executable.</p> <p><i>[Spend some time describing all the options in this window.]</i></p>	<p>Click the button labeled "Build" in the Project Manager.</p> <p>Click the "Cancel" button.</p>
Data Dictionary	<p>Let's first take a look at the Database Designer.</p> <p>As you can see, it starts out as an empty window. You'll also notice that a new menu option, "Database", is available. Using this menu, we can add a table, add a view, edit stored procedures, add indexes, and add referential integrity.</p> <p>We also have a toolbar associated with the database designer which has the options available to use.</p> <p>Notice that if we right-mouse click within the database designer, we also receive the same list of options. In Visual FoxPro, there is typically more than one way to do just about anything.</p> <p>Let's go ahead and make a table.</p> <p>In this example, I will use the Table Wizard.</p> <p><i>[If necessary, take a minute to explain how wizards work]</i></p> <p>In this dialog box we get to choose our own fields from a number of predefined sample tables. For this example, I will choose some fields from the Friends table.</p>	<p>In the Command Window, type "Close All".</p> <p>In the Command Window, type "Create Database mydbc".</p> <p>In the Command Window, type "Modify Database".</p> <p>From the "View" menu option, select "Toolbars..."</p> <p>From the list of toolbars, check the "Database Designer" and click "OK".</p> <p>Right-mouse click inside the Database Designer</p> <p>Right-mouse click inside the Database Designer and select "New Table".</p> <p>Select Table Wizard</p> <p>From the Sample Tables ListBox, select "Friends".</p> <p>Double-click the following fields from the Available Field ListBox: FriendID, FirstName, LastName, Nickname, Address, City, and State.</p>

Feature	Narrative	Keystrokes
	<p>This next dialog box allows us to change any aspect about each of the fields we selected. We can change the field name, the caption, the type, width, or number of decimals. In our example, let's change the FriendID to Numeric with a width of 5.</p> <p>The next dialog box allows us to set indexes on our table. For now, let's set the primary index to FriendID.</p> <p>The last dialog box asks us what we would like to do after the table is created. The options are nothing, browse the table, or modify the table in the Table Designer. We will leave it on the default option and click "Finish".</p> <p>As you can see, we now have a database in our database designer called "Friends".</p> <p>Let's now look at the Visual FoxPro's Table Designer. I can invoke it by right-mouse clicking on our new table and selecting "Modify...".</p> <p>[Take a minute to explain the options available in the Table Designer.]</p> <p>For an example, let's add a validation rule to our LastName field. Let's add "Not Empty (LastName)" for the rule and "Please supply a Last Name" as the text to be shown if the rule is violated.</p> <p>Now let's browse our table.</p>	<p>Click the "Next" Button.</p> <p>Change FriendID to Numeric with a width of 5.</p> <p>Click the "Next" Button.</p> <p>Set Primary key to "FriendID". Click the "Next" Button.</p> <p>Click the button labeled "Finish". In the save dialog box, name the database "Friends". (<i>Instructor's Note: Save this in the blank directory that you're working in.</i>)</p> <p>Right-mouse click on the Friends table and select "Modify...".</p> <p>Select "LastName" from the Table ListBox.</p> <p>Enter the following text in the TextBox next to Validation Rule: "NOT EMPTY(lastname)".</p> <p>Enter the following text (with quotes) in the TextBox next to Validation Text: "Please supply a first name".</p> <p>Click "OK".</p> <p>Click "Yes".</p> <p>Right-mouse click on our Friends database and select "Browse".</p>

Feature	Narrative	Keystrokes
	<p>I will now press Control-Y in order to add a new record.</p> <p>As you can see, Visual FoxPro won't add a new record because it violates our validation rule. How do you suppose we can get around this situation?</p> <p>Let's go back into our Table Designer and use the "Default Value" option to set a default last name. Let's use "Smith".</p> <p>Now lets try adding a new record.</p> <p>As you can see, Visual FoxPro added the record and set the Last Name field to Smith. But is this a practical solution? Not really. Another solution is to use what is called Data Buffering.</p> <p><i>[Take a minute to explain the different types of data buffering.]</i></p> <p>I will now turn on pessimistic row buffering.</p> <p>I will turn off my default value.</p> <p>Now let's try to add a blank record.</p> <p>As you can see, this solution is a lot cleaner. But what happens if we close the browse windows and attempt to close the database.</p> <p>As you'd expect, as soon as FoxPro tries to write our data buffers to disk, we get a warning that our validation rule has been violated.</p> <p>I am going to chose the Revert option. This will return the database to the state it was in prior to the violation.</p>	<p>Press CTRL-Y.</p> <p>Click "OK". Right-mouse click on the Friends database and select Modify to go back into the Table Designer. Enter the following text (with quotes) in the TextBox next to Default Value: "Smith". <i>(Instructor's Note: Include the quotes.)</i> Click "OK". Click "Yes". Right-mouse click on our Friends database and select "Browse".</p> <p>Press CTRL-Y.</p> <p>In the Command Window, type <code>"=CursorSetProp &#9633; ("buffering",2)"</code></p> <p>Return to the Table Designer and clear out the default value for last name. Open up the Browse window. Press CTRL-Y.</p> <p>Close the Browse window. In the Command Window, type: "Close Database".</p> <p>Click "Revert".</p>

Feature	Narrative	Keystrokes
	<p>If we go back into the Browse window. You can see that there are currently no records. Our newly created record was deleted in order to maintain our database's integrity.</p> <p><i>[Optional: you can add a valid record to this table to show that a valid record will not cause any validation errors.]</i></p> <p>Let's now close this database and open our sample database again. This is a sample database that is included with Visual FoxPro.</p> <p>Let's open a table.</p> <p>As you can see we have a number of fields defined. If I click the Index Tab, you will see we have four indexes defined as well. With an index, you can define a name, type, expression, and filter. <i>[Take a moment to explain type, Expression and Filter].</i></p> <p>We also have table properties. As you can see we can set table level validation rules, set up triggers for Inserts, Updates, and Deletes, and even supply table comments.</p> <p>Let's now take a minute to look at how we set up "persistent relationships" between tables. If you look at our customer table and orders table, you will see a line connecting the CUST_ID index from the customer table to the CUST_ID index of the order table. This line indicates that a persistent relationship has been set up between to the two tables. Because there is a plus sign on side of the line and a "crows foot" on the other, we know that this is a one to many relationship.</p> <p>These relationships were set up back in the analysis stage of the development process.</p>	<p>Return to the Browse window.</p> <p>In the Command Window, type: "Close All". From the File menu, select Open. Open the file TESTDATA.DBC located in the SAMPLES\DATA directory in your Visual FoxPro directory. Remember to select the database type from the file type combo box..</p> <p>Right-mouse click on the Customer table and select Modify.</p> <p>Click the Index Tab.</p> <p>Select the button labeled "Table Properties ...".</p> <p>Close the window Click the button labeled "Cancel".</p> <p>Use the mouse to select the line in between the customer and order table.</p>

Feature	Narrative	Keystrokes
	<p>Someone determined that there needed to be a relationship between customers and orders. <i>[If necessary, more detail regarding relationships and the analysis stage may be added here.]</i></p> <p>Let's show how we can set up a persistent relationship. First I am going to delete the one that I have currently selected.</p> <p>Now I am going to click the CUST_ID field from the customer table and drag it to the CUST_ID field of the order table.</p> <p>A dialog box appears and ascertains that a one-to-many index should be assigned. This is correct so we click "OK".</p> <p>That's it! We now have a relationship setup between the two tables.</p> <p>Another setting available to us is referential integrity.</p> <p>First, you'll notice that I have a grid that lists all persistent relationships within this database container. <i>[Take a minute to talk about the rule options available for the Update, Delete, and Insert triggers]</i>. In this example, let's set the Delete rule to "Restrict" and the Insert rule to "Restrict". This will prevent us from deleting a customer if orders exist and will prevent us from adding orders to a customer that doesn't exist.</p> <p>Now I will select OK.</p> <p>Answer "Yes" to saving changes.</p> <p>We now have a message that informs us that it will attempt to merge new RI code with any old RI code. We will respond "Yes".</p>	<p>Press the Delete key.</p> <p>Click and drag the CUST_ID text from the customer table to the CUST_ID text of the order table.</p> <p><i>(Instructor's Note: Customer Primary Key to Orders Index.)</i></p> <p><i>(Note: All relationships in the Data Designer are index to index.)</i></p> <p>Click the button labeled "OK".</p> <p>Right-mouse click on our persistent relationship line and select "Referential Integrity" from our context sensitive menu.</p> <p>For the top row, select "Restrict" from the ComboBox in the Delete column and Insert column.</p> <p>Click "OK".</p> <p>Click "Yes".</p> <p>Click "Yes".</p>

Feature	Narrative	Keystrokes
	<p>In order to see this new RI code, let's open our stored procedure code for this database container.</p> <p>This shows you all the code that was added by selecting those two RI rules.</p> <p>If I return to the Table Designer.</p> <p>And go into the table properties.</p> <p>You can see that I now have triggers defined that point to lines of code within my table's stored procedures. <i>[Optional: Go into the table properties of the Orders table and show that it has code references as well.]</i></p>	<p>Right-mouse click anywhere on the background within the Database Designer. Select "Stored Procedures..." from the context sensitive menu.</p> <p>Scroll down though the window and then close it.</p> <p>Right-mouse click in the customer table and select "Modify...".</p> <p>Click the "Table Properties" button.</p> <p>Close the window. Click "Cancel."</p> <p>In the Command Window, type: Close All</p> <p>In the Command Window, type: Clear All</p>

Feature	Narrative	Keystrokes
Form Designer	<p>Let's start by running through the Form Wizard.</p> <p>The first task that needs to be done is to locate a table to associate the wizard with. In this case, I will use the Customer table from our sample database.</p> <p>Now we select some fields that we wish to expose to the user.</p> <p>I then click on "Next".</p> <p>There are five options as to the style of our form. We could select a standard style, or a chiseled, shadowed, boxed, or embossed style as well. For this example, let's choose embossed.</p> <p>The form will also contain navigation buttons by default. You can choose between text buttons, picture buttons, or even no buttons if you wish to program them yourselves. For this example, I will select "picture" buttons.</p>	<p>Select "New" from the "File" menu. Click the radio button "Form" followed by the "Wizard" button. Click "Form Wizard" and click "OK".</p> <p>Click the ellipses ('...') and locate the file CUSTOMER.DBF in the SAMPLES\DATA directory.</p> <p>Select the following fields: Cust_id, Company, Address, City, Postalcode, Phone.</p> <p>Click the button labeled "Next".</p> <p>Select "Embossed"</p> <p>Select Picture Buttons.</p> <p>Click the button labeled "Next".</p>

Feature	Narrative	Keystrokes
	<p>This next dialog will let you set up a sort order for your data. In this example, we will select City and Company.</p> <p>That's it. We now have the option to give our form a title. I will leave the default name, "Customer". You will also notice a new button called "Preview". This will let us preview our form before saving it. Let's preview our form.</p> <p>As you can see, the form looks just as we would have expected. We can now click the "Return to Wizard" button to return to the wizard.</p> <p>Since we like the form, I will select the option "Save form and modify it in the Form Designer" and click "Finish."</p> <p>We will give it the default name of "customer".</p> <p>That's it. We now have our form ready to go.</p> <p>We will return to this form, but for the next example, I am going to close this particular example down.</p> <p>I am now going to create a blank form.</p> <p>I now have a blank form on the screen. Let's also open up the Form Controls toolbar.</p>	<p>Select City followed by Company. Click the button labeled "<u>N</u>ext".</p> <p>Click the button labeled "<u>P</u>review".</p> <p>Click the button labeled "<u>R</u>eturn to Wizard".</p> <p>Click the option "<u>S</u>ave form and <u>m</u>odify it in the Form Designer".</p> <p>Click "<u>F</u>inish".</p> <p>Click the button labeled "<u>S</u>ave".</p> <p>Close the Form Designer.</p> <p>In the Command Window, type: MODI FORM TEST1</p> <p>Select the "<u>F</u>orm Controls Toolbar" option under the "<u>V</u>iew" menu. <i>(Instructor's Note: These may already be available on the screen.)</i></p>

Feature	Narrative	Keystrokes
	<p><i>[At this point, the instructor is advised to place one of each of the following controls on the form: TextBox, ComboBox, Label, CommandButton, Edit Box, Option Group, Check Box, Spinners, Images, PageFrames, and ListBox. Take a minute to talk about each control and use the Properties window to demonstrate some of these features.]</i></p> <p>Let's now take a look at the Data Environment.</p> <p>The Data Environment window is what we can use to specify the tables that are used by our form. To add a table, right-mouse click inside the Data Environment and select "Add."</p> <p>Normally, if we have a database container open at this time, the tables for that would be listed in the ListBox. In this case, we will open a table by selecting the "Other" button.</p> <p>We now have a table associated with this form. A nice feature we can use at this time is dragging fields onto our form.</p> <p>Another nice feature provided to us is the ability to align controls.</p> <p>If I select my four fields, I can use my layout toolbar to align control exactly how I'd like them.</p> <p><i>[Take some time and demonstrate the option available within the Layout toolbar using these four fields].</i></p>	<p>Close all windows.</p> <p>In the Command Window, type: MODI FORM TEST2 Right-mouse click inside the new form and select "Data Environment" from the context sensitive menu.</p> <p>Right-mouse click inside the Data Environment and select "Add..." from the context sensitive menu.</p> <p>Click the "Other..." button. Locate the CUSTOMER.DBF in the \SAMPLES\DATA directory.</p> <p>Click and drag four fields onto the form. <i>(Pay no attention to where on the form they go).</i></p> <p>Select the "Layout Toolbar" option from the "View" menu. <i>(Instructor's Note: These may already be available on the screen.)</i></p>

Feature	Narrative	Keystrokes
	<p>Now let's run this form. We can do that by clicking the toolbar button with an exclamation point on it.</p> <p>As you can see, we have live, editable data already in our Textboxes.</p> <p>Let's add a button to the form.</p> <p>Let's have this button move the record pointer.</p> <p>The first line is a standard Xbase command that skips the record pointer forward one record. The next line, which we will go into with more detail later on, refreshes the form and updates the Textboxes with the new record. Let's execute this form.</p> <p>Let's change the caption on our button to say "Next".</p> <p>Now lets add three more buttons to our form and name them, "Prev", "Save", and "Cancel".</p> <p>Now lets add some code to their Click events.</p>	<p>Click the toolbar button with an exclamation point. (<i>Instructor's Note: If the toolbar isn't available, right-click on the form and select run.</i>)</p> <p>Close the window.</p> <p>In the Command Window, type: MODI FORM TEST2</p> <p>Add a CommandButton anywhere on the form and then double-click it.</p> <p>Enter the following text into the code window: SKIP IN CUSTOMER THISFORM.REFRESH()</p> <p>Click the toolbar button with an exclamation point. (<i>Instructor's Note: If the toolbar isn't available, right-click on the form and select run.</i>)</p> <p>Click the CommandButton a few times.</p> <p>Close the window. In the Command Window, type: MODI FORM TEST2</p> <p>Change the caption property on the button by using its properties sheet to say "Next".</p> <p>Add three more buttons with the captions "Prev", "Save", and "Cancel".</p> <p>Double-click the "Prev" button and add the following text: SKIP -1 IN CUSTOMER THISFORM.REFRESH()</p> <p>Double-click the "Save" button and add the following text: =TABLEUPDATE()</p>

Feature	Narrative	Keystrokes
---------	-----------	------------

Feature	Narrative	Keystrokes
	<p>Lastly, we need to set the data buffering.</p> <p>Now let's execute this form again.</p> <p>Let's now return to the Data Environment and link in a second table.</p> <p>I'm going to add a second table—in this case our Orders table.</p> <p>As you can see, we have a pre-defined persistent relationship already setup. This is the relationship that connects orders to customers.</p> <p>Now I'm going to click and drag the entire Orders table onto my form.</p> <p>When you drag an entire table, the FoxPro Form Designer uses a Grid control in order to hold all the fields plus data. Now we'll run this application.</p> <p>[Demonstrate form]</p> <p>On last trick I'd like to show regarding this form. I am going to go into the Form Properties...</p> <p>And I'll select the Data tab...</p>	<p>Double-click the "Cancel" button and add the following text: <code>=TABLEREVERT()</code> <code>THISFORM.REFRESH()</code></p> <p>Open the Data Environment, right-click on the table, and select "Properties".</p> <p>Change the property "BufferModeOverride" to 2.</p> <p>Click the toolbar button with an exclamation point.</p> <p>[Demonstrate the new buttons]</p> <p>Open the Data Environment</p> <p>Add the Orders Table.</p> <p><i>(Instructor's Note: In order to make room to add a grid control, make sure that the four Textboxes are located in the upper fourth on the form and the buttons are in the lower fourth.)</i></p> <p>Click and drag the entire Orders table onto the center of the form. You may need to resize it in order to prevent it from overlapping other controls.</p> <p>Execute the form.</p> <p>Close the form</p> <p>In the Command Window, type: <code>Modify Form Test2</code> Go into the Form Properties. Select the Data Tab.</p>

Feature	Narrative	Keystrokes
	<p>Now I'm going to execute my form again.</p> <p>I am now going to execute two more instances of my form.</p> <p>I now have three forms that can each interrogate the data independently.</p> <p><i>[Feel free to conduct other examples if time exists.]</i></p>	<p>Change the property DataSession to "2 - Private Data Session"</p> <p>Execute Form.</p> <p>In the Command Window, type: DO FORM TEST2 .SCX DO FORM TEST2 .SCX</p> <p>Set focus to each form and move the record point to show that the forms are independent.</p>
Understanding Events	<p>To help you understand events, I'd like to run a small sample application that comes with Visual FoxPro that shows you all the events that fire based on certain controls.</p> <p>As you can see, what may seem like a simple event may actually be a series of four or five events. This example is a good example to use if you are curious about the order in which events can occur and if you're unsure if or when an event gets triggered.</p> <p><i>[Demonstrate the kind of events you can generate by manipulating many of the sample controls.]</i></p>	<p>From the Command Window, type: DO FORM EVENTS .SCX from the directory: \\SAMPLES\\CONTROLS\\EVENTS\\</p> <p><i>[Demonstrate different events]</i></p>
The Grid Control	<p>Let's take a closer look at the Grid control. I'm going to start fresh with a new form within the Form Designer.</p> <p>Now I will drag the Grid onto my form.</p> <p>One nice feature in Visual FoxPro is Builders. A Builder is similar to a Wizard except that they manipulate already created objects and can be used over and over. If I right-click the grid, I see an option for a Builder inside the context sensitive help.</p>	<p>In the Command Window, type: MODI FORM TEST3 Add the Employee table to the Data Environment.</p> <p>Drag the Employee table onto the Form to create a Grid.</p> <p>Right-mouse click on Grid and select "Builder."</p>


Feature	Narrative	Keystrokes
	<p>If I select the Grid Builder, I now have a wizard-like application that will guide me through all the changes I can make to my grid. The first tab asks us to select the fields we wish to use for this Grid.</p> <p>The next tab allows us to select a style for the grid. Let's select "Professional"</p> <p>The next tab allows us to adjust the column width in order to best display our data.</p> <p>Since we are only working with one table, this tab is not necessary. Normally, this tab will allow you to set up one-to-many relationships between tables. For now, let's click "OK".</p> <p>See how easy that was to set up?</p> <p><i>[If time is available, demonstrate some of the other properties available with the Grid control].</i></p>	<p>Select fields: Emp_id, Last_name, First_name, and Home_phone.</p> <p>Click tab "2. Style"</p> <p>Select "Professional" Click tab "3. Layout"</p> <p>Adjust column width. Click tab "4. Relationship"</p> <p>Click "OK". Execute the form.</p>
The PageFrames Control	<p>There are many times when your form simply has too many controls on it. Microsoft has provided us with an excellent control to combat this problem, it is called PageFrames. If you have worked with other GUI development languages, you might have heard this called a Tab control as well. Let's take a look at a simple example.</p> <p>By default, we are given two tabs. By manipulating the properties of this control, we are able to change the number of tabs, color, font, or style. To add a control to the tab, we must first bring that tab into focus. We can do this by using the properties window.</p>	<p>In the Command Window, type: MODI FORM TEST4 Add a PageFrame control to the form.</p>

Feature	Narrative	Keystrokes
	<p>You should now see a thick GotFocus line surrounding the PageFrame control. This means we are ready to add controls to Tab #1.</p> <p>Now let's add some to the second page.</p> <p>We can also add controls that appear on all pages. For example, let's add a button that will appear on all pages.</p> <p>Let's execute this to see how it looks.</p> <p><i>[If time is available, demonstrate some of the other properties available with the PageFrame control].</i></p>	<p>Open the properties window and select "Page1" in the Object ComboBox.</p> <p>Add a ComboBox and CommandButton to the PageFrame.</p> <p>Select "Page2" in the Object ComboBox of the properties window.</p> <p>Add a ListBox and CheckBox to PageFrame.</p> <p>Click mouse outside of PageFrame to remove focus.</p> <p>Add a CommandButton to the lower right corner of PageFrame.</p> <p>Execute Form.</p> <p>When you're done, type: CLOSE ALL CLEAR ALL</p>
Using Views	<p>There are times in every Xbase application where the developer would like to display a certain subset of the data to your user based on multiple tables and multiple criteria. In Visual FoxPro, this becomes very easy using FoxPro views.</p> <p>Let's suppose that we wanted to design a view that would show all customers with overdue orders. How would we do it?</p> <p>First let's open our Testdata database container. We can use the right-mouse click to add a new view to our container.</p> <p>Now we have a blank view. In order to accomplish our task, we need to use the Customer table.</p>	<p>Open the TestData database container.</p> <p>Right-mouse click and select "New Local view..."</p> <p>Click "New View"</p> <p>Select Customer table and click "OK".</p>

Feature	Narrative	Keystrokes
	<p>What I could do now is select the fields I need...</p> <p>and browse this new view.</p> <p>At this point, you can see that our new view looks exactly like our other tables. In actually, FoxPro will allow us to treat our view as if it were a table called "MyView".</p> <p>If we browse the table, we can see all our customers. Now this isn't exactly what we wanted, is it? No. I wanted a list of all customers with past-due orders. Let's return to the View Designer.</p> <p>To complete this, I need to add another table to this view. Let's add the orders table.</p> <p>Notice that our view has already picked up a persistent relationship between the customer table and the orders table. You can also see an entry in the "Selection Criteria" grid that mirrors this persistent relationship. I'm going to add another criteria here. I am going to add a criteria that states that the field "require_by" in the orders table...</p> <p>... is less than...</p> <p>10 days ago.</p> <p>I need one more criteria now. I am going to say that the "Shipped on" field...</p> <p>must equal</p>	<p>Click the "Fields" Tab. Select "cust_id" and "company" fields.</p> <p>Close Window. Select "Yes" Call the view, "MyView"</p> <p>Right-mouse click and select "Browse".</p> <p>Close Browse window Right-mouse click on MyView and select "Modify...".</p> <p>Add Orders table.</p> <p>Click selection criteria tab.</p> <p>Under Field Name, select "Orders.require_by".</p> <p>Under Criteria select "Less Than"</p> <p>Under the Example, type: DATE () - 10</p> <p>Under Field Name, select "Orders.shipped_on".</p> <p>Under Criteria select "Equal"</p>

Feature	Narrative	Keystrokes
	<p>an Empty date.</p> <p>What my criteria now says is that only records that have the "require_by" field less than 10 days ago and that have the "shipped_on" date as empty can be in this table.</p> <p>One more thing I need to do is set my group order. Since some customers have more than one order, it would help in clarity to group each of the orders based on the customer.</p> <p>I will select the customer ID to sort on. Now let's browse the table.</p> <p>As you can see, we now have a smaller set of data than our last browse.</p> <p>We now have our list of customers with past due orders. Let's suppose that we wanted to include a grid that contained those past due orders. We will need to create another view based on the same criteria as the previous plus another criteria based on the selected record in the first table.</p> <p>Let's create our view.</p> <p>As you can see, this is similar to our first view. I will now add this new criteria.</p>	<p>Under the example, type: { / / }</p> <p>Click the "Group By" tab. Select the "cust_id" field as the Group By field.</p> <p>Close the window. Browse the table.</p> <p>From the Database Designer, add a new local view. Use the Orders table. Add the following criteria: Orders.require_by Less Than DATE() - 10 and Orders.shipped_on Equal { / / }</p> <p>Add the following criteria: Orders.cust_id Equal ?lngCustID</p>

Feature	Narrative	Keystrokes
	<p>What do you suppose the question mark is for in our criteria? The question mark gets used when we want to define a parameter during runtime. Let's see what happens when we browse this.</p> <p>As you can see, because we didn't supply a value for our lngCustID parameter, FoxPro prompts us for it. I will enter "BOTTM" because I know this is a record with a past due order.</p> <p>Now we have our past due records.</p> <p>Let's now create a form that we can use to better display this information.</p> <p>Now we have our two view associated with our form. Now I haven't placed any controls on my form yet, what do you suppose will happen if I run it?</p> <p>Bingo! We get hit with a message asking for a value in our parameter, lngCustID. Obviously, this is not what we want our users to see. Visual FoxPro provides a property we can use to fix this.</p> <p>The property is called "NoDataOnLoad". For this example, let's change that property to True.</p>	<p>Select Fields Tab. Select the following fields: cust_id, to_name, order_date, shipped_on. Select Order by Tab. Select order_date as ordering criteria. Right-click and Select Run query.</p> <p>Enter "BOTTM"</p> <p>Close the Browse window. Close the View Window. Save it with the name PASTDUE.</p> <p>In the Command Wwindow, type: CLOSE ALL CLEAR ALL MODI FORM PASTDUE Open the form's Data Environment. Open the Database TESTDATA from the <u>F</u>ile menu. Click and drag our two views into our Data Environment window. Close the Database Designer.</p> <p>Execute Form. <i>[If you didn't receive the parameter missing error, type "Close All" and "Clear All" in the Command Window]</i></p> <p>Right-mouse click on our second view in the Data Environment.</p> <p>Change "NoDataOnLoad" property to True.</p>

Feature	Narrative	Keystrokes
	<p>Now let's add a ComboBox and execute its builder.</p> <p>I will now select our TestData database container.</p> <p>We do not want our users to be able to add items to the ComboBox so lets change the selection to a "Drop-down List".</p> <p>Now we have a ComboBox. Now I want to add a Grid control for our orders. As you remember from before, the easiest way to do this is to click and drag our view from our Data Environment right onto our form.</p> <p>Unfortunately, we do need to add some code to get this to work. I am now going to go into the code window under the Form - Init event procedure.</p> <p>I am now going to refresh the grid prior to the form loading.</p> <p>I am now going to our ComboBox control. Inside the event "InteractiveChange", I also want to refresh the grid.</p> <p>Lastly, we add code to our Grid control.</p>	<p><i>(Instructor's Note: Supply a parameter such as BOTTM if asked.)</i></p> <p>Add a ComboBox to the form and start its builder by right-clicking on the builder.</p> <p>Click the "... " button and locate the TESTDATA.DBC file. Select the first view we created. Select both the cust_id and company fields. Click the tab "2 - Style".</p> <p>Change radio buttons to "Drop-down List". Click "OK".</p> <p>Click and drag our second view onto our form to create a Grid.</p> <p>Open the code window. <i>(Instructor's Note: Choose Code from the View Menu.)</i> Open the Form - Init procedure. <i>(Instructor's Note: Use the combo boxes on the left and right to navigate.)</i></p> <p>In the code window, type: <code>THISFORM.GRID1.REFRESH()</code></p> <p>Open the Combo - InteractiveChange procedure. In the code window, type: <code>THISFORM.GRID1.REFRESH()</code></p> <p>Open the Grid1 - Refresh procedure. In the code window, type: <code>lngCustID =</code>  <code>THISFORM.COMBO1.VALUE</code> <code>=Requery("pastdue")</code> <i>[Substitute your second view's name in the requery if different].</i></p>

	<p>Now let's execute it.</p> <p>That is how easy it is to work with views.</p>	<p>Execute form and demonstrate.</p>
Using Reports	<p>We will now create a simple report using the Report Wizard.</p> <p>The first step in our wizard asks us to select a table to build from. I am going to choose the customer table.</p> <p>I will select the fields company, country, region, and maximum order amount.</p> <p>We now need to select our level of groupings. I will select a grouping on country and on region. Therefore, my report will initially be grouped by country and then sub-grouped by region.</p> <p>I'd also like a sub-total on each grouping and a master total at the end.</p> <p>Now I get to select the sort order within our sub group. I will select alphabetical ordering based on Company for my order.</p> <p>The next selection is style. For clarity in this demonstration, I will select ledger.</p> <p>Here's my chance to select a title for my report. I am going to use "Maximum Order Amount by Country".</p> <p>Now let's take a quick look at our report by using the preview button.</p> <p>[Show report to audience]</p>	<p>Select File-New. Select Report and click "Wizard". Select "Group/Total Report Wizard".</p> <p>Open the Customer table.</p> <p>Select fields "Company", "Country", "Region", and "Maxordamt". Click "Next".</p> <p>Select "Country" as first group. Select "Region" as second group.</p> <p>Select the "Totals" and Subtotals" Checkboxes. Click "Next".</p> <p>Select Company. Click "Next".</p> <p>Select "Ledger" Click "Next".</p> <p>Change title to: "Maximum Order Amount by Country".</p> <p>Click "Preview" button.</p>

Feature	Narrative	Keystrokes
	<p>[Take some time to explain some of the features within the Report Designer.]</p>	<p>Close preview window. Select "Save report and modify it in the Report Designer" option. Leave the default name and click "Save".</p>
Object-Oriented Programming	<p>When you are designing forms in Visual FoxPro, you may find yourself, at one time or another, cutting and pasting multiple controls onto a form. If you find yourself doing this, you might ask yourself if this control is a good candidate for being designed as a class. For example, I am going to create a form and create a square on top of it.</p> <p>Now I'm going to cut and paste my square three more times.</p> <p>Now I might say to myself, should I create a class for my yellow squares. Let's say I said, "Yes".</p> <p>Let's remove those squares now. I will now select my one square and under the File menu select "Save as Class".</p> <p>Once I do that, I receive a dialog asking me to describe my class. I will call this class "MyShape" and save it as "MyClass.vcx".</p> <p>Now I am going to remove my last yellow square as well. On the Forms Control Toolbar, there is a PushButton with some books on it. If I click it, and then click "Add", it will ask me which class do I wish to add to my toolbar.</p>	<p>Create a new form (call it Test5). Create a yellow square on the form (not too big because we need to fit three more squares on this form).</p> <p>Select the square. Press CTRL-C. Press CTRL-V three times and position the squares equally on the form.</p> <p>Delete those three new squares.</p> <p>Select the only yellow square. Click the "Save as Class" menu option under "File".</p> <p>Enter the name "MyShape". Enter the filename "MyClass.vcx" Click "OK".</p> <p>Delete the yellow square.</p>

Feature	Narrative	Keystrokes
	<p>You will notice that our toolbar of standard controls has now been replaced by a small toolbar that contains our one class, MyShape.</p> <p>I will now add MyShape to my form.</p> <p>Notice in the properties window that my new shape is named, "MyShape1".</p> <p>Let's add another MyShape.</p> <p>As predicted, this shape is named, "MyShape2".</p> <p>So what else can I do with my class? I'm now going to close and save this form.</p> <p>And from the File menu, select "Open"...</p> <p>and select my class file, "MyClass.vcx".</p> <p>Now I must select the class I'd like to edit. I will select "MyShape".</p> <p>I am now within my Class Designer. One of the properties of this shape is called curvature. Let's change that to make our shape more round. I'm also going to change my shape to red.</p> <p>Now I'm going to close down the Class designer and save my class. I will now execute the form I had just created.</p> <p>Look at that! I didn't touch my form, yet the new classes took effect on my form. [Use this moment to talk briefly about encapsulation.]</p>	<p>Click the "View Classes" button on the toolbar and click "Add". Select our class and click "Open".</p> <p>Place the cursor over the MyShape button for a second to allow the Tooltip to display.</p> <p>Add MyShape to the form.</p> <p>Show properties window.</p> <p>Add another MyShape to the form. Show properties window.</p> <p>Close and save form.</p> <p>Select "Open" from the "File" menu. (Instructor's Note: Be sure to have the Visual Classes file type in the file type dialog.) Select "MyClass.vcx" and click "OK".</p> <p>Select "MyShape".</p> <p>Change the curvature to 99. Change the background color to Red.</p> <p>Close and save MyClass. From the Command Window, type: DO FORM TEST5</p>

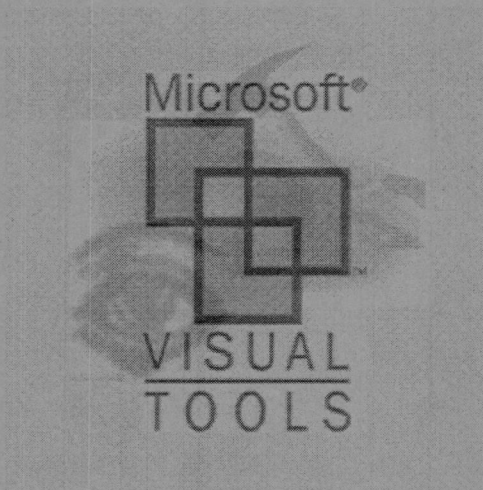
Feature	Narrative	Keystrokes
	<p>Now let's take a look at one of the predefined classes Visual FoxPro has included for us. I'm going to open the classes that Visual FoxPro uses for its wizards.</p> <p>As you can see, I have plenty of classes to choose from.</p> <p>Let's suppose the we wanted our class to perform some task. Well, one way is to go into our form designer and add the task there.</p> <p>Now if I click my shape...</p> <p>it turns to green. But what if I click my other shape. Nothing.</p> <p>Let's show a better way to do this. I'm now going back into my Class Designer and will open the MyShape class. I will now add this code to my class.</p> <p>Notice that I use the name "THIS" to describe my object. Using the "THIS" object name means that I don't care what name you give the object of this class. Change its properties to the color green. Now lets execute our form again.</p> <p>Now both my circles turn green. [Use this moment to talk briefly about polymorphism.]</p>	<p>Close the form.</p> <p>Select "Open" from the "File" menu. Open the file "WizStyle.vcx" in the "Wizards" directory.</p> <p>Cancel window.</p> <p>Open Test5 in the Form Designer. Double-click one of the circles. Add the following code: THISFORM.MYSHAPE1 . ↵ BACKCOLOR = RGB(0,255,0) Execute form.</p> <p>Click shape.</p> <p>Close form. Open Test5 in the Form Designer. Remove code that was added. Close Designer. (Instructor's Note: Go back to the Class Designer.)</p> <p>Double-click circle. Add the following code: THIS.BACKCOLOR = ↵ RGB(0,255,0)</p> <p>Close and save MyClass. From the Command Window, type: DO FORM TEST5</p> <p>[Note: if the first circle does not turn green, then you left code within its click event. Delete it and retry.]</p>

Feature	Narrative	Keystrokes
	<p>OK. Now that we have defined this event in our class, what if we would like one of our objects on our form to do something different. Well, lets go back into our Form Designer and add something to the click event.</p> <p>As you can see, if I click the object that isn't subclassed, it turns green. If I click the object I modified, it displays the MessageBox but does not turn green. <i>[Use this moment to talk briefly again about subclassing and inheritance.]</i></p> <p>Well, let's say that I wanted it to turn green as well. Visual FoxPro allows us special syntax that allows us to call the classes event as well as any event code we wanted to add.</p> <p>There! As you can see, we have complete control over our objects.</p> <p><i>[If the instructor wishes, they can continue with this example to discuss topics like "Adding Properties" or "Adding Methods".]</i></p>	<p>Open Test5 in the Form Designer. Double-click one of the circles. <i>(Instructor's Note: Make sure you have the click event showing in the code window.)</i></p> <p>Add the following code: <code>=MessageBox("This is subclassed")</code> Execute form.</p> <p>Close the form. Open Test5 in the Form Designer. Double-click the circle with code. Add the following code at the bottom: <code>MYSHAPE: :CLICK()</code> Execute form.</p>
Putting It All Together	<p>Let's start by creating a project file.</p> <p>Now I'd like to add a menu to my project.</p> <p>I am now in the Menu Designer. I will now add a menu option called "Stuff".</p> <p><i>[Explain the different Result options.]</i> I will select "submenu".</p>	<p>Create a project called "MyProj".</p> <p>From the Project Manager, select "Menu" and click the "New" button.</p> <p>Type "Stuff" as the first prompt.</p> <p>Select "submenu" and click "create".</p>

Feature	Narrative	Keystrokes
	<p>My submenu will have two options on it, "Hello"...</p> <p>And "Goodbye".</p> <p>I will now close my menu as save it as "MainMenu".</p> <p>I'm now going to build an application...</p> <p>And execute it.</p> <p>As you can see, I have my menu and my two options. Why do you suppose I have a Format menu as well? This format menu is actually associated with the Command Window. This means that my application ended. Keep in mind that the menu does not have a wait state. If this was an executable, the application will display the menu and then instantly terminate. What do we have to do to cure this? Add a wait state.</p> <p>I will now add a program to my project that first calls the menu.</p> <p>And then I will call a Visual FoxPro command call READ EVENTS.</p> <p>This places Visual FoxPro into a wait state.</p> <p>I now need to tell my project that I need to run this code instead of my menu.</p>	<p>Type "Hello" as the first prompt.</p> <p>Select "command". In the textbox, add: WAIT WINDOW "Hello"</p> <p>Type "Goodbye" as the second prompt. Select "command". In the textbox, add: DO FORM GOODBYE</p> <p>Close Menu Designer and save menu as "MainMenu".</p> <p>Build application and ignore all errors.</p> <p>Execute application.</p> <p>In the Command Window, type: SET SYSMENU TO DEFAULT In the Project Manager, select "programs" and click "New"</p> <p>Add the following code: DO MAINMENU.MPR</p> <p>Add the following code: READ EVENTS</p> <p>Save code as "Startup.prg"</p> <p>Select the "Startup" line in the Project Manager and from the "Project" menu, select "Set Main".</p>

Feature	Narrative	Keystrokes
	<p>As you can see, this places a black dot next to the code indicating that it is the starting module. Now let's add a menu option that gets us out of the event loop:</p> <p>Once again I will build the application.</p> <p>As you can see, the Command Window is gone and our two menu options are all that's available.</p> <p><i>[If time is available, the instructor can also demonstrate creating an EXE with this project.]</i></p> <p><i>[If time is available, the instructor can also execute the Setup wizard in order to show how distribution disks are made.]</i></p>	<p>Double-click the "mainmenu" line in the Project Manager.</p> <p>Add a Menu option called "Quit".</p> <p>Select "Command".</p> <p>Add the code "CLEAR EVENTS".</p> <p>Close the Menu Designer.</p> <p>Build Application and ignore errors.</p> <p>Execute Project.</p>

.....



**Microsoft® Developer
Seminar-in-a-Box Series**

**Visual FoxPro 3.0
Seminar Slides**



Legal Disclaimer and copyright slide

Please remove this slide prior to
presenting, but leave it in for any
electronic or material
photocopying or distribution

Microsoft, Windows, and Win32, are registered trademarks and Visual FoxPro is a trademark of Microsoft Corporation. All other trademarks, marked and not marked, are property of their respective owners.

This document is provided for informational purposes only. The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to change in market conditions, it should not be interpreted to be a commitment on the part of Microsoft and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

INFORMATION PROVIDED IN THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND FREEDOM FROM INFRINGEMENT. The user assumes the entire risk as to the accuracy and the use of this document. This document may be copied and distributed subject to the following conditions: 1) All text must be copied without modification (except foreign language translation) and all pages must be included; 2) All copies must contain Microsoft's and Application Developer Training Company's copyright notice and any other notices provided therein; and 3) **This document may not be distributed within the boundaries of Canada or the United States of America.**

Copyright © 1996 Application Developers Training Company. All Rights Reserved.

Microsoft Product Number: 098-65270

Visual FoxPro 3.0



Suggestions for the beginning of this seminar:

- Welcome the audience.
- Introduce yourself.
- Let the audience know approximately how long the seminar will last, and when they can expect to take breaks.
- Tell the audience where they can find the restrooms and telephones.
- If there are designated smoking areas, indicate where they are.

All set--then let's go!

The New Visual FoxPro

♦ Design Tools

- Improved Project Manager
- Form Designer
- Report Writer Enhancements
- Class Designer
- Database Designer



Discuss briefly the new features of Visual FoxPro and what you will cover in this seminar. This, and the next two slides, provide an overview and are simply to provide structure to the seminar.

Guide the audience through Chapter One in the workbook.

Visual FoxPro is a cross-platform development environment. You may simultaneously develop for Windows 3.1, Windows 95, Windows NT, or for the Macintosh. Visual FoxPro for Unix and DOS operating systems have not been announced.

Assure the audience that the applications that they developed in FoxPro 2.x are not obsolete, and that they can migrate their current applications to Visual FoxPro and keep developing.

The New Visual FoxPro

◆ Event-Model

- Visual FoxPro is better integrated with Windows and responds to most Windows events.
- Visual FoxPro objects respond to events like:
 - ◆ Activate/Deactivate
 - ◆ Resize
 - ◆ Refresh
 - ◆ And many others...



Discuss the events that FoxPro 2.x could respond to, and compare that with the events that you can respond to in Visual FoxPro.

Begin Visual FoxPro, and start up the Taz Trader sample application.

The New Visual FoxPro

- ◆ Object-Oriented Programming (OOP)
 - Built-in Classes
 - Support for Properties, Events, and Methods
 - Support for Encapsulation, Polymorphism, and Inheritance



Briefly mention that Visual FoxPro is a fully-OOP development environment, and that you'll spend more time discussing OOP later in the seminar.

Starting with the Project Manager

- ◆ Creating a project
- ◆ Organizing your development environment
- ◆ Adding files to the Project Manager
- ◆ Configuring the Project Manager



Discuss the functionality of the Project Manager. Points to cover:

- Outline format
- Organization of the tabs
- Docking the Project Manager
- Tearing off Project Manager tabs

Discuss the elements of a project.

Discuss the elements of organizing applications: directories, conventions used, etc..

Talk about the configuration of a project using the Project Info dialog.

Points to cover:

- Developer information
- Debug code being added to project
- Encrypting project code
- Setting an icon for the project

Do Demonstration: Project Manager

Using the Data Dictionary

- ◆ The purpose of a Data Dictionary
- ◆ Adding tables and rules to your Data Dictionary
- ◆ Creating relationships between tables
 - Enforcing referential integrity
- ◆ Other database features
 - Triggers
 - Buffering



Discuss the elements of the Data Dictionary. Points to cover:

- What is it?
- Design surfaces
- Tool bars

Discuss the validation of keys and business rules using the data dictionary. Points to cover:

- Primary, Candidate and regular keys
- Default values
- Field and table level validation
- Referential Integrity
- Triggers

Do Demonstration: Data Dictionary

Working with Forms

- ◆ Starting with the Form Wizard
- ◆ Working in the Form Designer
- ◆ Form Designer features:
 - Properties sheet
 - Form Controls toolbar
 - Layout toolbar
 - Color palette
 - Data Environment



Introduce the concept of forms and talk about the Form Wizard.

(Optional: If there many FoxPro 2.x developers in your audience, compare and contrast to FoxPro 2.x screens and the FoxPro Power Tools.)

Talk about the tools that are integrated with the Form Designer. Key points:

- Property sheet
- Form Controls toolbar
- Layout toolbar
- Color palette
- Data Environment

Do Demonstration: Form Designer

Working with Forms

◆ Form Design Terms

- Controls/Objects
- Properties
- Events
- Methods

◆ Understanding Events

◆ Using Properties and Methods in Code

- Object.Property
- Object.Method



Discuss the different design terms used by Visual FoxPro. Talk about the importance of properties and how to set them.

Discuss the concept of events. Talk about how they relate to Windows, what triggers them, and how we assemble code that handles these events.

Discuss the syntax Visual FoxPro uses to assign properties and calling methods in code. Be sure to talk about Visual FoxPro's system variables like ActiveControl, THIS, THISFORM, and THISFORMSET and why they are important to know.

Be sure to include a discussion on the Refresh method and why it is important in Visual FoxPro.

Do Demonstration: Understanding Events

Working with Forms

♦ Visual FoxPro's Enhanced Controls

- Grids
- PageFrames (Tabbed Dialogs)
- Timers
- Combo boxes
- Custom toolbars
- OLE



Discuss some of the enhanced controls available to the Visual FoxPro developer.

Do Demonstration: The Grid Control

Do Demonstration: The PageFrame Control

Creating Output

- ◆ Understanding Views
 - Local vs. remote views
- ◆ Using the View Wizard
- ◆ Modifying views with the View Designer
- ◆ Accessing Client/Server data with Visual FoxPro



Discuss the concepts of creating report output.

Begin by talking about views and the View Wizard.

Expand on the capabilities of views by discussing:

- The View Designer
- Parameterized Views
- Extending views to client/server with remote views.

Discuss the various features in the View Designer. In particular, the function of all the tabs in the View Designer.

Discuss how to create parameterized views and why they are important in an application.

Do Demonstration: Using Views

Formatting Your Output

- ◆ Understanding the importance of formatted output
- ◆ The Report Wizard
- ◆ The Report Designer
 - The Data Environment
 - Report Properties



Discuss how you can take the output from a view and format it using the Report Wizard.

After exploring the Report Wizard you can then go into the Report Designer and talk about its capabilities.

Points to cover:

- Data Environment
- Layout toolbar works here too
- Adding objects to reports
- Grouping and Formatting

Do Demonstration: Using Reports

Object-Oriented Programming in Visual FoxPro

♦ Terminology and Concepts

- Classes
- Properties
- Methods
- Encapsulation
- Inheritance
- Polymorphism
- Subclassing



This section will discuss OOP at as high a level as possible.
Begin the discussion by talking about the new terminology.

Object-Oriented Programming in Visual FoxPro

- ◆ The importance of creating re-usable components
- ◆ Establishing a firm foundation
 - Foundation classes
 - Power of subclassing



Discuss the benefits of creating re-usable components. Point out:

- Data hiding
- Project Modularization
- Code re-usability

Talk about the importance of having a good set of foundation classes.
Point out the need for planning ahead when designing your classes.

Talk about the concept of primitive and the benefits involved with being
able to ripple changes in the parent class to all child classes.

Working in the Class Designer

- ◆ Creating your own classes
- ◆ Where to begin
- ◆ Using the Class Designer
 - Adding custom properties/methods
 - Specifying class information
- ◆ Understanding the power of inheritance
- ◆ Adding custom components to your development environment



Discuss the Class Designer and discuss its features in detail:

- Adding properties and method to classes.
- Specifying class code and properties.
- Power of inheritance by sub-classing.

Show how you can use controls in your applications by adding your custom class library to the Form Designer toolbar.

Do Demonstration: Object-Oriented Programming

Putting the Application Together

- ◆ Application distribution concepts
- ◆ Creating an application menu
- ◆ Creating a main program
- ◆ Building distributable applications
- ◆ Using the Setup Wizard



This section focuses on distributing an application.

Discuss how to create a menu with the Menu Designer.

Create a main program with the code shown in the book.

Note that READ EVENTS places Visual FoxPro in a wait state for processing user menu requests.

Discuss the differences between creating an .APP and creating an .EXE using the Project Manager.

If time allows you may want to discuss using the Setup Wizard.

Do Demonstration: Putting it all together

Adding OLE

- ◆ The importance of OLE to Visual FoxPro developers
- ◆ Understanding OLE terminology
- ◆ Using OLE components in Visual FoxPro
 - Bound controls
 - In-place editing
 - OLE Automation
 - OLE Custom Controls



Discuss the aspects of OLE and its purpose.

Topics to include:

- General fields and bound controls
- Unbound controls and in-place editing. Excel provides good examples for this.
- Using OLE Automation. Create a conversation with Word or Excel to demonstrate this.
- Using OLE custom controls. Discuss the power of component development.

Do Demonstration: OLE

Visual FoxPro and the Internet

- ◆ Why the Internet is important to Visual FoxPro developers.
- ◆ Using the Visual FoxPro Internet Wizard
- ◆ Components of the Internet Wizard
 - Search Page Wizard
 - Visual FoxPro WWW server



Note: This section is optional and is not covered in great detail in the workbook.

Discuss the emergence of the Internet

Discuss the features of the VFP Internet Wizard:

- Search page creator.
- Server application.

Demonstrate the Search Page Wizard.

Load the Server application.

Send and receive data from the HTML page generated. This is going to require you to use Microsoft Internet Information server or some other server on NT.